US006185619B1

(12) **United States Patent**
Joffe et al.

(10) **Patent No.: US 6,185,619 B1**
(45) **Date of Patent: Feb. 6, 2001**

(54) **METHOD AND APPARATUS FOR BALANCING THE PROCESS LOAD ON NETWORK SERVERS ACCORDING TO NETWORK AND SERVE BASED POLICIES**

(75) Inventors: **Rodney Lance Joffe**; **Barry A. Dykes**, both of Phoenix, AZ (US); **Jason Alan Brittain**, Fairfield, CA (US); **Victor Joseph Oppleman**, Phoenix, AZ (US); **Brian Everett Pettingell**, Phoenix, AZ (US); **James Joseph Lippard**, Phoenix, AZ (US); **Ian Burke Vandeventer**, Tempe, AZ (US); **Brett Dean Watson**, Beaverton, OR (US); **Steven Michael Hotz**, Marina del Rey, CA (US); **Nils Herbert Mc Carthy**, Phoenix, AZ (US)

(73) Assignees: **Genuity Inc.**, Burlington, MA (US); **GTE Service Corporation**, Irving, TX (US)

(*) Notice: Under 35 U.S.C. 154(b), the term of this patent shall be extended for 0 days.

(21) Appl. No.: **08/965,848**

(22) Filed: **Nov. 7, 1997**

Related U.S. Application Data

(60) Provisional application No. 60/032,484, filed on Dec. 9, 1996.

(51) **Int. Cl.**^{7} **G06F 15/16**

(52) **U.S. Cl.** **709/229**; 709/223; 709/225

(58) **Field of Search** 395/200.68, 675, 395/200.53, 200.62, 200.8; 709/225, 226, 229, 250, 232, 220, 223

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,399,531	*	8/1983	Grande et al.	370/60
4,587,651		5/1986	Nelson et al.	370/88
4,771,424	*	9/1988	Suzuki et al.	370/86
4,873,517		10/1989	Baratz et al.	340/825.03
4,958,341		9/1990	Hemmady et al.	370/60.1
5,218,601	*	6/1993	Chujo et al.	370/228
5,231,631		7/1993	Buhrke et al.	370/60
5,278,829		1/1994	Dunlap	370/94.1
5,323,394		6/1994	Perlman	370/85.13
5,329,623		7/1994	Smith et al.	395/275
5,341,477	*	8/1994	Pitkin et al.	395/200.56
5,347,633		9/1994	Ashfield et al.	395/200
5,351,237		9/1994	Shinohara et al.	370/58.3
5,361,259		11/1994	Hunt et al.	370/84

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

0648038A	12/1995	(EP)	H04L/29/06

OTHER PUBLICATIONS

"Load Balancing for Multiple Interfaces for Transmission Control Protocol/Internet Protocol for VM/MVS", *IBM Technical Disclosure Bulletin,* vol. 38, No. 9, Sep. 1, 1995, pp. 7–9, XP000540166.

European Search Report, PCT Application No. PCT/US 97/22542.

Office Stack Quick Start Guide, No Publication Date.

Office Stack Installatin Guice (1994).

Multi–Tenant System Manager Manual, No Publication Date.

Intellicom Ethernet Switching (1994).

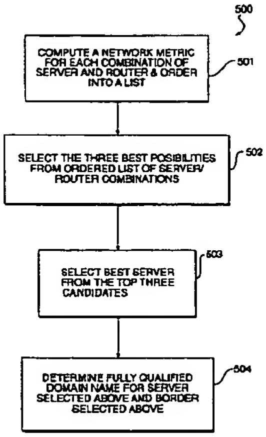
Side Winder Products Document (1995).

Primary Examiner—Zarni Maung
(74) *Attorney, Agent, or Firm*—Leonard Charles Suchyta

(57) **ABSTRACT**

According to the present invention, a method and system provides the ability to assign requests for data objects made by clients among multiple network servers. The invention provides a distributed computing system and methods to assign user requests to replicated servers contained by the distributed computing system in a manner that attempts to meet the goals of a particular routing policy. Policies may include minimizing the amount of time for the request to be completed.

29 Claims, 15 Drawing Sheets


500
COMPUTE A NETWORK METRIC FOR EACH COMBINATION OF SERVER AND ROUTER & ORDER INTO A LIST
501
SELECT THE THREE BEST POSIBILITIES FROM ORDERED LIST OF SERVER/ ROUTER COMBINATIONS
502
SELECT BEST SERVER FROM THE TOP THREE CANDIDATES
503
DETERMINE FULLY QUALIFIED DOMAIN NAME FOR SERVER SELECTED ABOVE AND BORDER SELECTED ABOVE
504

U.S. PATENT DOCUMENTS

5,398,012		3/1995	Derby et al.	340/825.03
5,404,451		4/1995	Nemirovsky et al.	395/200
5,414,698		5/1995	Adams	370/17
5,416,842		5/1995	Aziz	380/30
5,422,878		6/1995	Kimoto et al.	370/60
5,434,863		7/1995	Onishi et al.	370/85.13
5,436,902		7/1995	McNamara et al.	370/85.3
5,442,630		8/1995	Gagliardi et al.	370/85.13
5,444,782		8/1995	Adams, Jr. et al.	380/49
5,452,294		9/1995	Natarajan	370/54
5,452,330		9/1995	Goldstein	375/257
5,455,826		10/1995	Ozveren et al.	370/60
5,459,720		10/1995	Iliev et al.	370/60
5,475,685		12/1995	Garris et al.	370/82
5,485,455		1/1996	Dobbins et al.	370/60
5,490,252		2/1996	Macera et al.	395/200.01
5,491,694		2/1996	Oliver et al.	370/85.4
5,499,297		3/1996	Boebert	380/23
5,517,620		5/1996	Hashimoto et al.	395/200.15
5,519,836		5/1996	Gawlick et al.	395/200.15
5,521,910		5/1996	Matthews	370/54
5,539,883		7/1996	Allon et al.	395/200.11
5,544,169		8/1996	Norizuki et al.	370/60.1
5,546,452		8/1996	Andrews et al.	379/219
5,548,533		8/1996	Gao et al.	364/514 C
5,572,643	*	11/1996	Judson	395/200.48
5,581,552	*	12/1996	Civanlar et al.	370/396
5,596,719		1/1997	Ramakrishnan et al.	395/200.02
5,600,794		2/1997	Callon	395/200.01
5,603,029	*	2/1997	Aman et al.	395/675
5,627,971	*	5/1997	Miernik	709/200
5,644,713		7/1997	Makishima	395/200.01
5,646,943		7/1997	Elwalid	370/230
5,774,660	*	6/1998	Brendel et al.	395/200.31
5,774,668	*	6/1998	Choquier et al.	395/200.53
5,828,847	*	10/1998	Gehr et al.	395/200.69
5,867,706	*	2/1999	Martin et al.	395/675

* cited by examiner

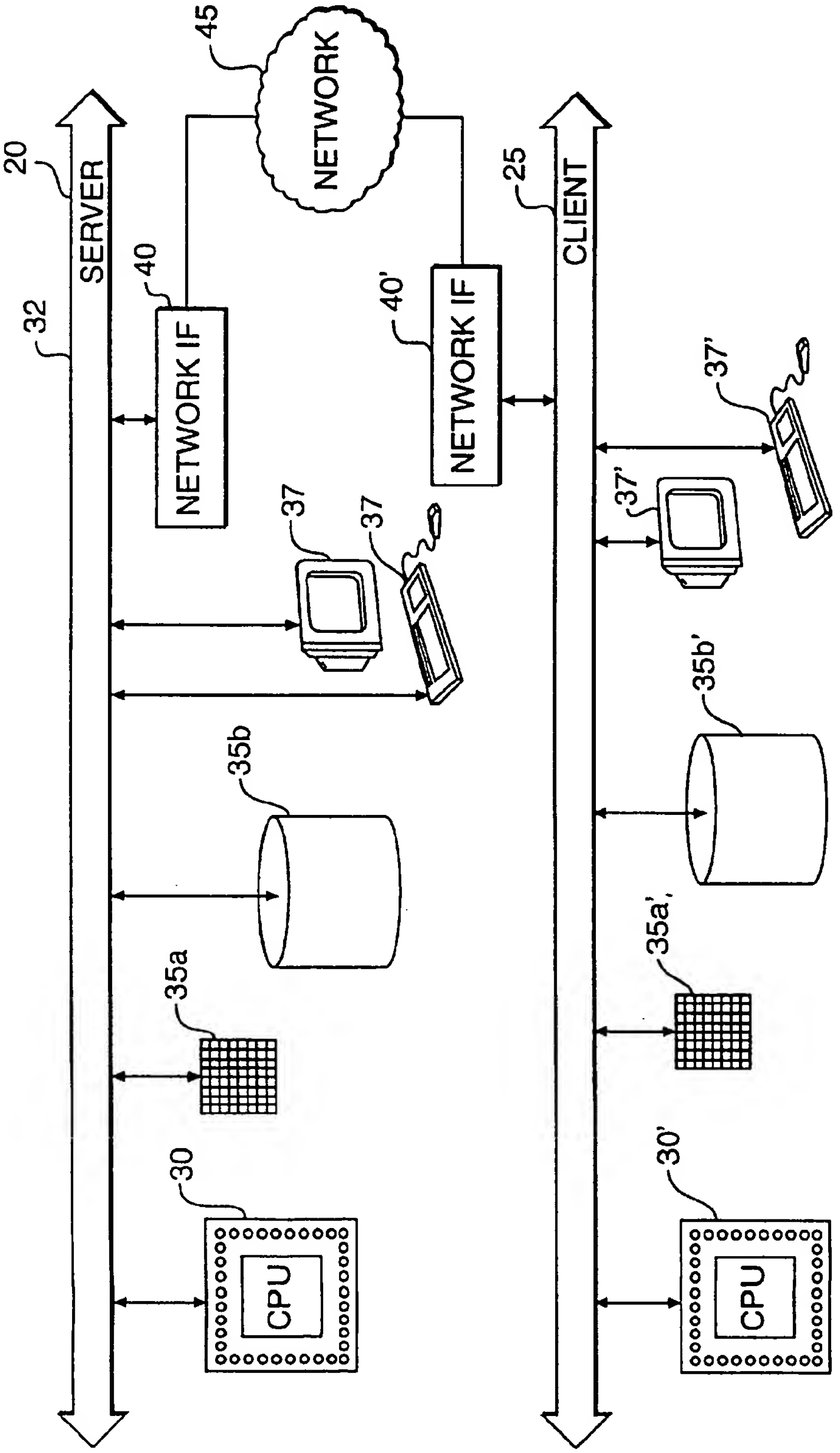


FIG. 1A
(PRIOR ART)

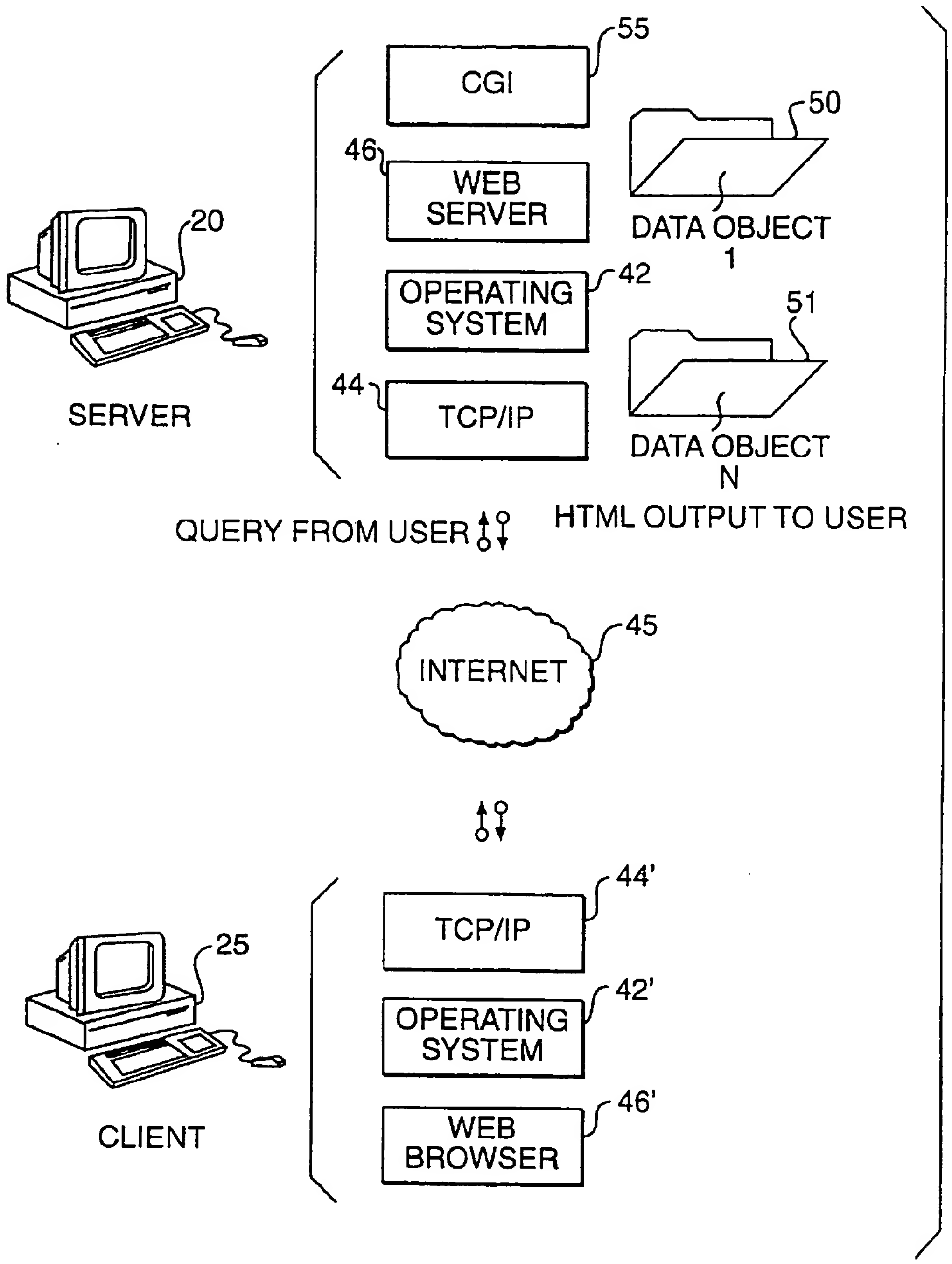


FIG. 1B
(PRIOR ART)

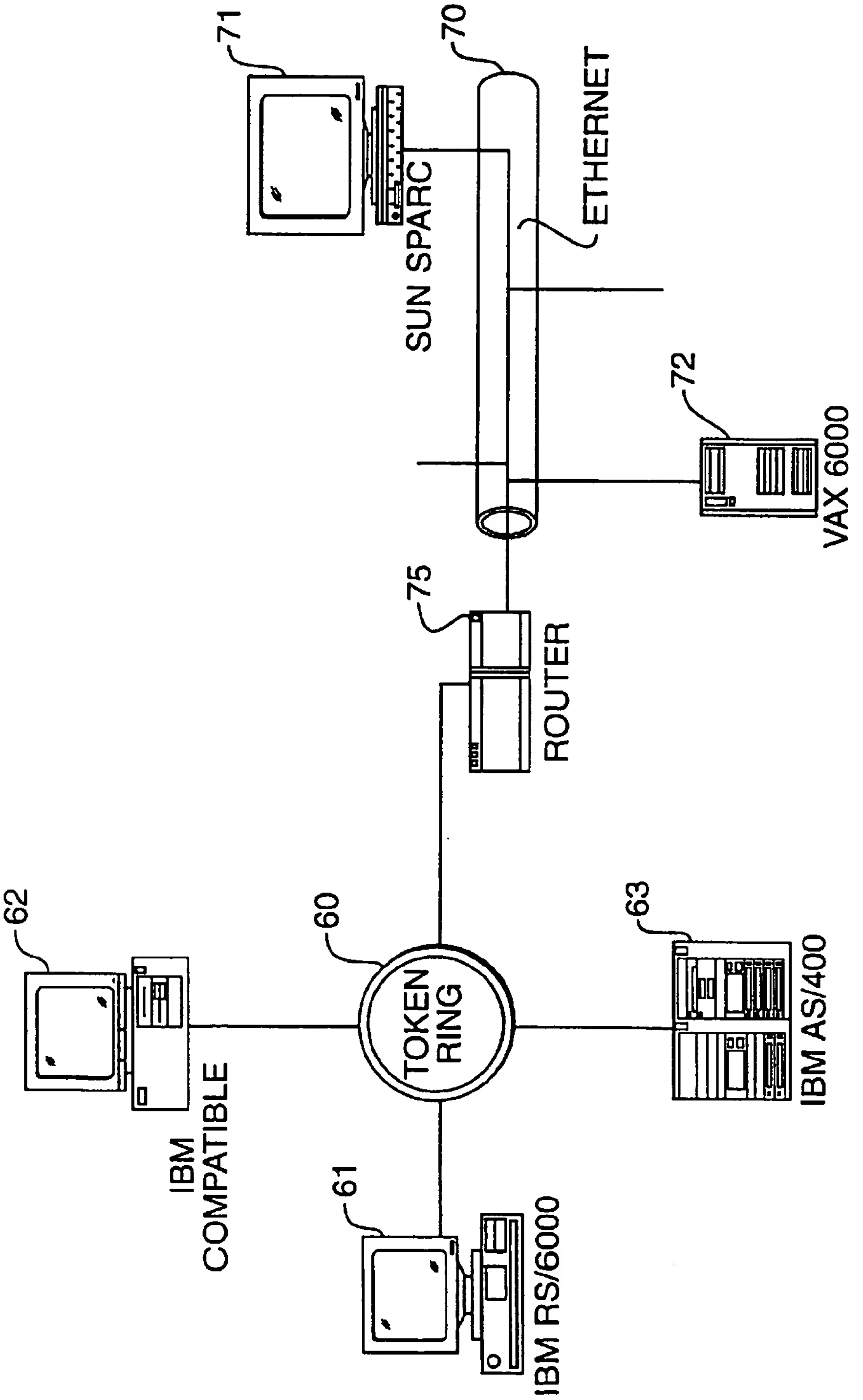
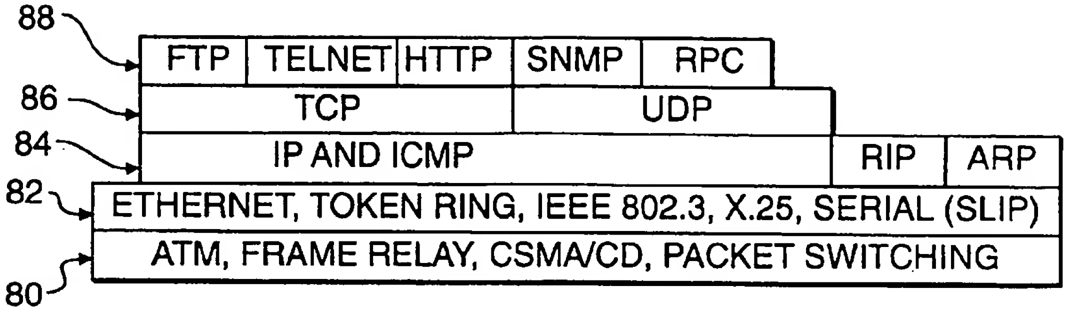


FIG. 1C
(PRIOR ART)


88
FTP
TELNET
HTTP
SNMP
RPC
86
TCP
UDP
84
IP AND ICMP
RIP
ARP
82
ETHERNET, TOKEN RING, IEEE 802.3, X.25, SERIAL (SLIP)
ATM, FRAME RELAY, CSMA/CD, PACKET SWITCHING
80

LEGEND

88	SESSION/APPLICATION LAYER
86	TRANSPORT LAYER
84	NETWORK LAYER
82	DATA LINK LAYER
80	PHYSICAL LAYER

FIG. 1D
(PRIOR ART)

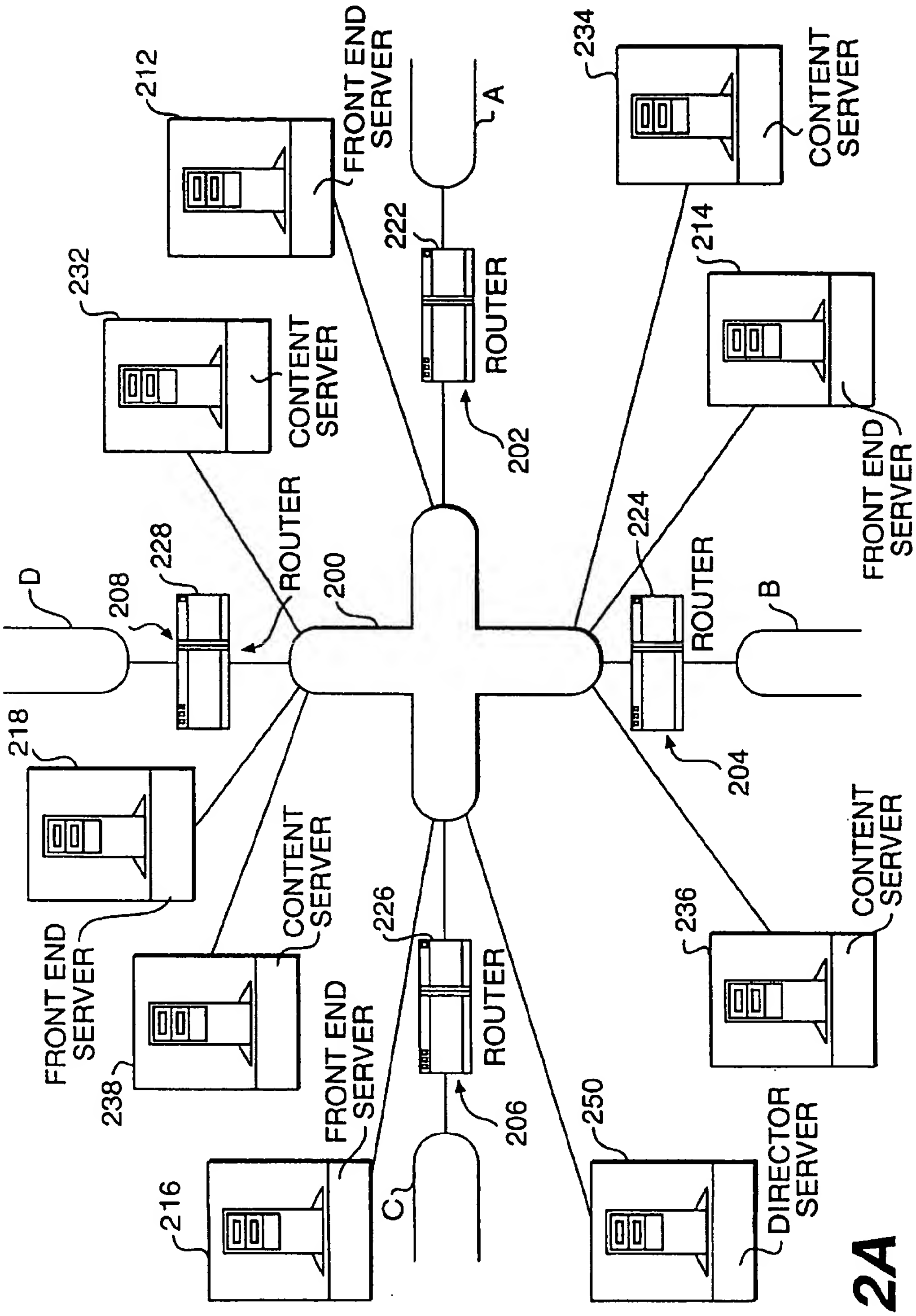

212
FRONT END SERVER
A
234
CONTENT SERVER
232
CONTENT SERVER
222
ROUTER
202
214
FRONT END SERVER
D
208
228
ROUTER
200
224
ROUTER
B
204
218
CONTENT SERVER
FRONT END SERVER
238
226
ROUTER
206
236
CONTENT SERVER
FRONT END SERVER
216
C
250
DIRECTOR SERVER

FIG. 2A

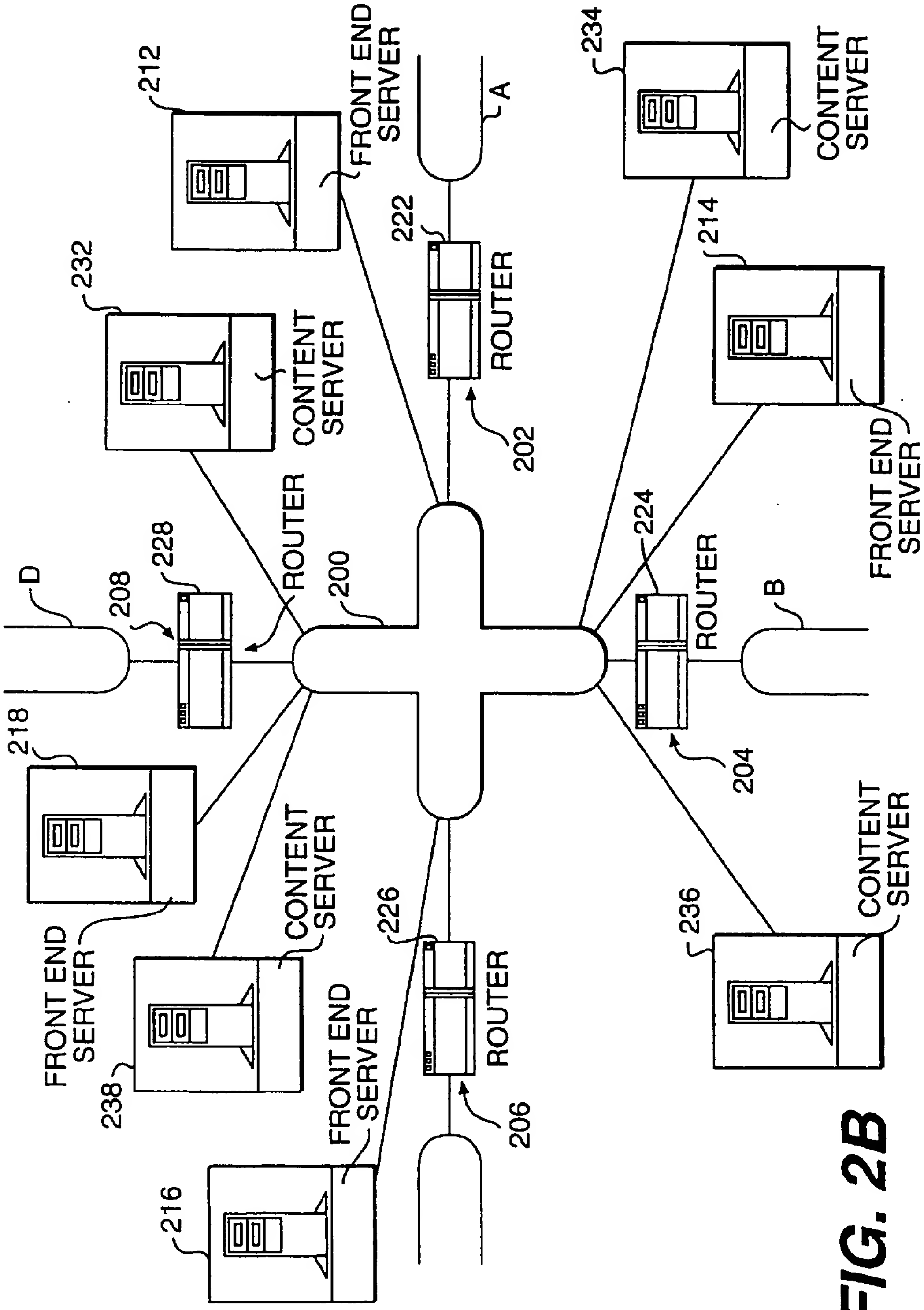

200
202
ROUTER
222
A
212
FRONT END SERVER
232
CONTENT SERVER
228
ROUTER
208
D
218
FRONT END SERVER
238
CONTENT SERVER
216
FRONT END SERVER
226
ROUTER
206
234
CONTENT SERVER
214
FRONT END SERVER
224
ROUTER
204
B
236
CONTENT SERVER

FIG. 2B

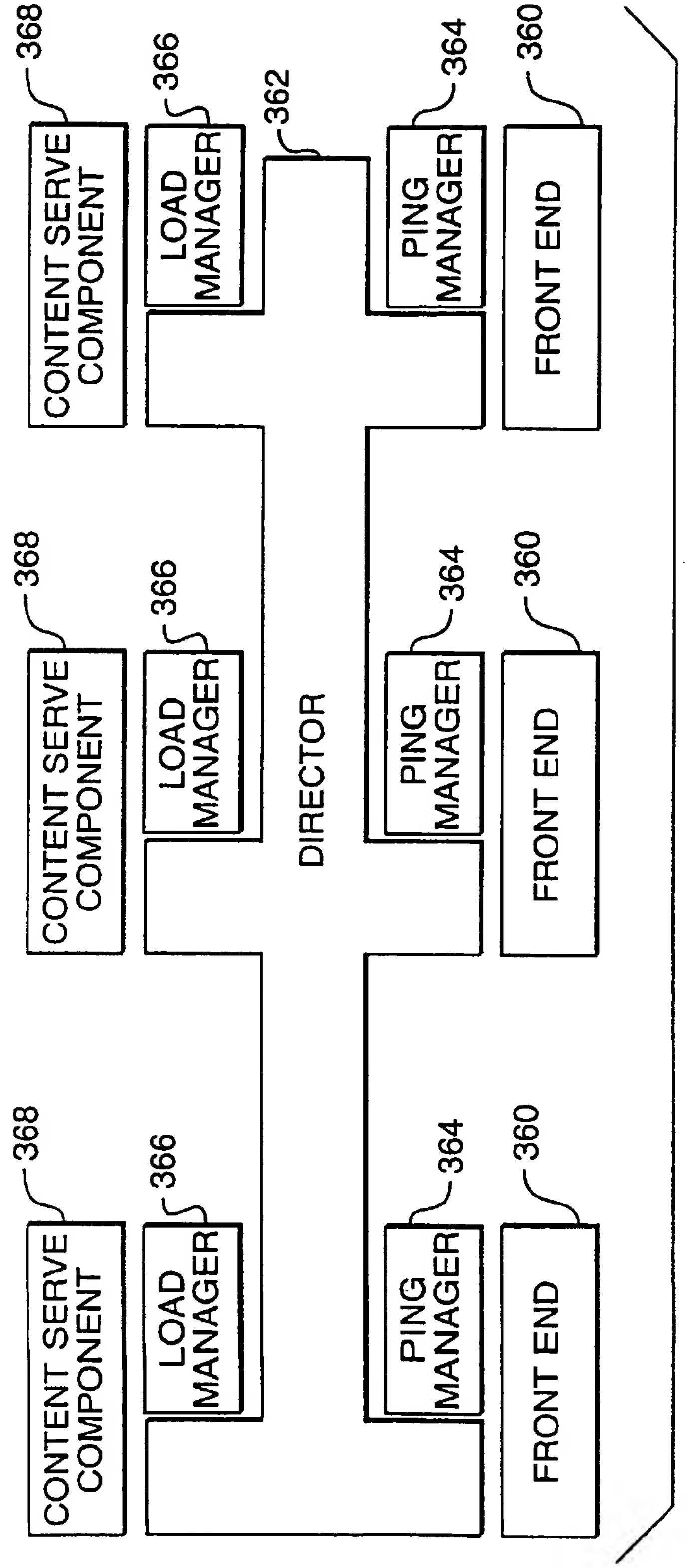

CONTENT SERVE COMPONENT
368
LOAD MANAGER
366
362
PING MANAGER
364
FRONT END
360
CONTENT SERVE COMPONENT
368
LOAD MANAGER
366
DIRECTOR
PING MANAGER
364
FRONT END
360
CONTENT SERVE COMPONENT
368
LOAD MANAGER
366
PING MANAGER
364
FRONT END
360

FIG. 3A

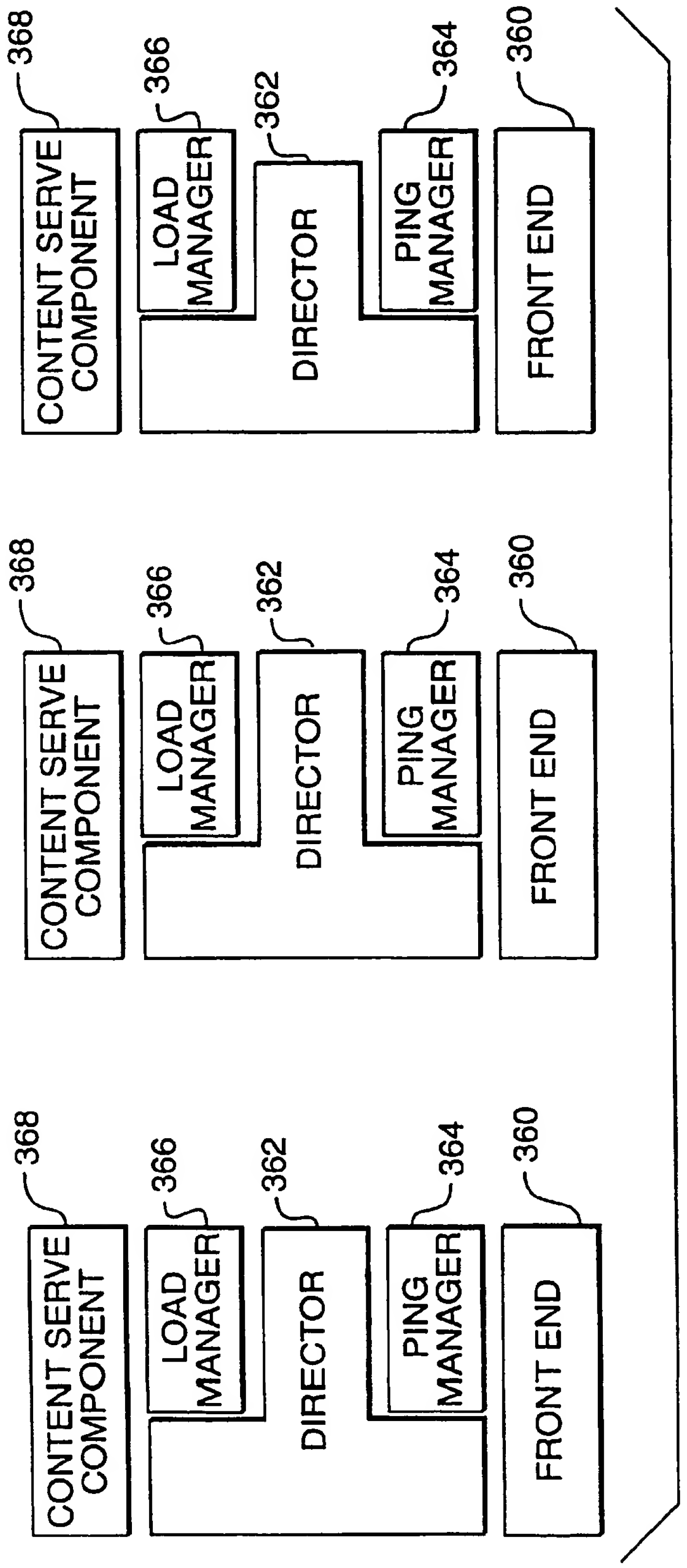

CONTENT SERVE COMPONENT
368
LOAD MANAGER
366
DIRECTOR
362
PING MANAGER
364
FRONT END
360
CONTENT SERVE COMPONENT
368
LOAD MANAGER
366
DIRECTOR
362
PING MANAGER
364
FRONT END
360
CONTENT SERVE COMPONENT
368
LOAD MANAGER
366
DIRECTOR
362
PING MANAGER
364
FRONT END
360

FIG. 3B

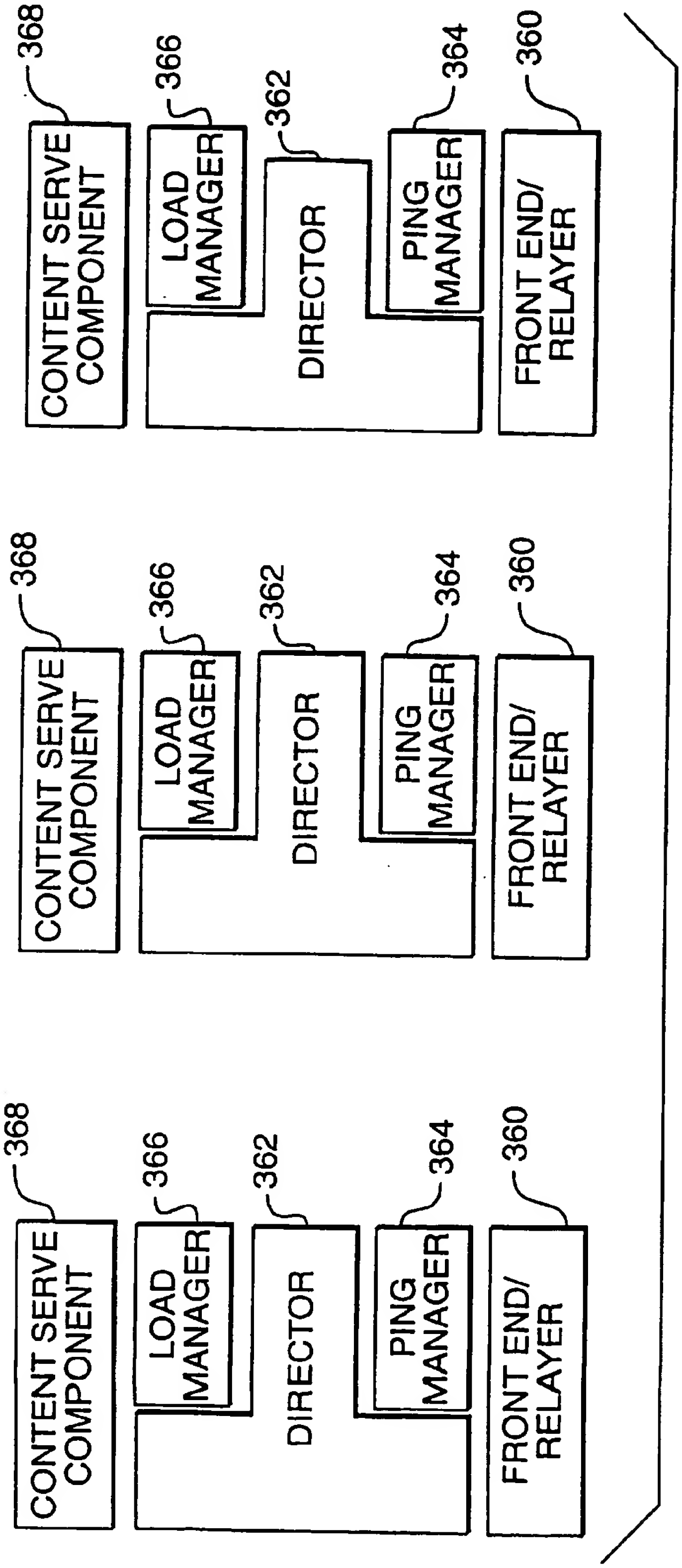

368
CONTENT SERVE COMPONENT
366
LOAD MANAGER
362
DIRECTOR
364
PING MANAGER
360
FRONT END/ RELAYER
368
CONTENT SERVE COMPONENT
366
LOAD MANAGER
362
DIRECTOR
364
PING MANAGER
360
FRONT END/ RELAYER
368
CONTENT SERVE COMPONENT
366
LOAD MANAGER
362
DIRECTOR
364
PING MANAGER
360
FRONT END/ RELAYER

FIG. 3C

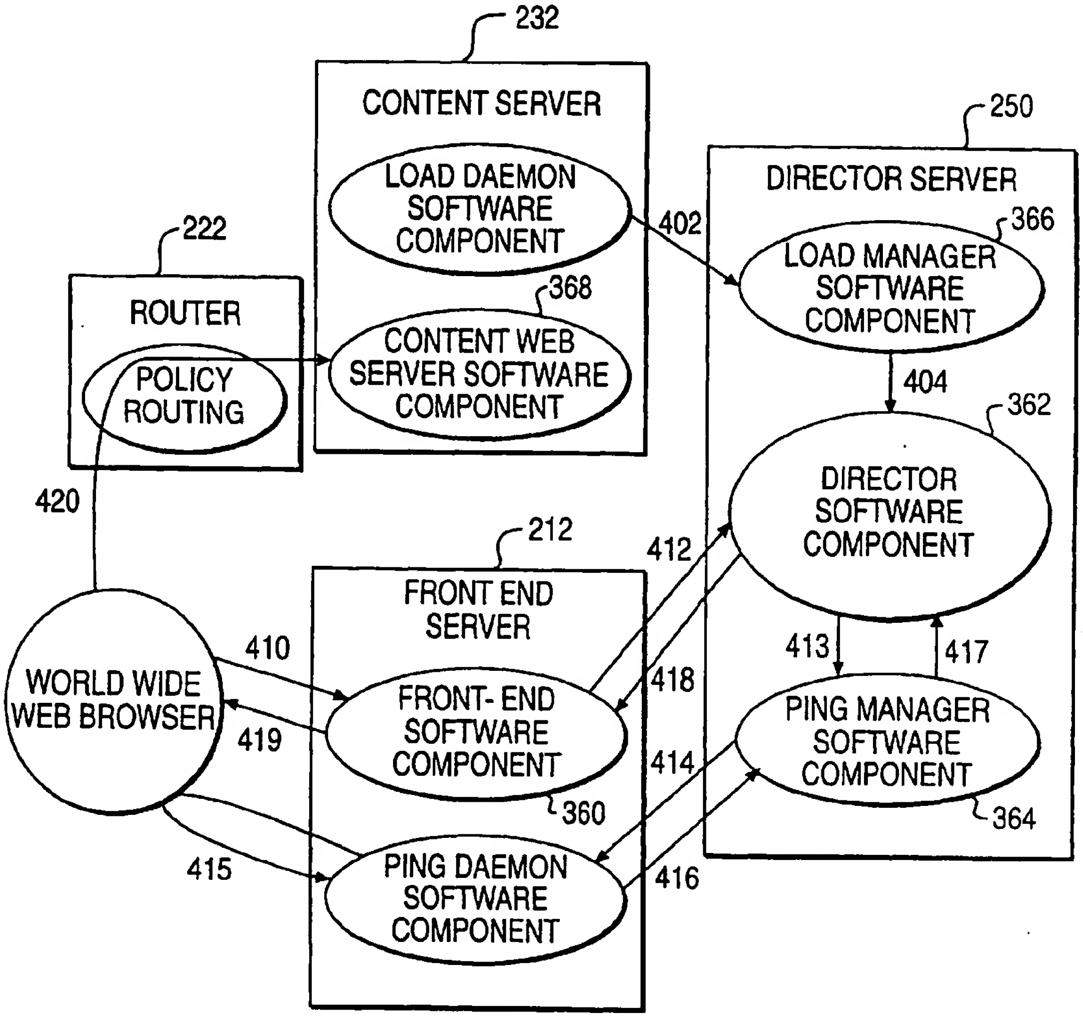

232
CONTENT SERVER
LOAD DAEMON SOFTWARE COMPONENT
368
CONTENT WEB SERVER SOFTWARE COMPONENT
222
ROUTER
POLICY ROUTING
420
250
DIRECTOR SERVER
366
LOAD MANAGER SOFTWARE COMPONENT
402
404
362
DIRECTOR SOFTWARE COMPONENT
412
418
413
417
PING MANAGER SOFTWARE COMPONENT
364
414
416
212
FRONT END SERVER
FRONT- END SOFTWARE COMPONENT
360
PING DAEMON SOFTWARE COMPONENT
WORLD WIDE WEB BROWSER
410
419
415

FIG. 4A

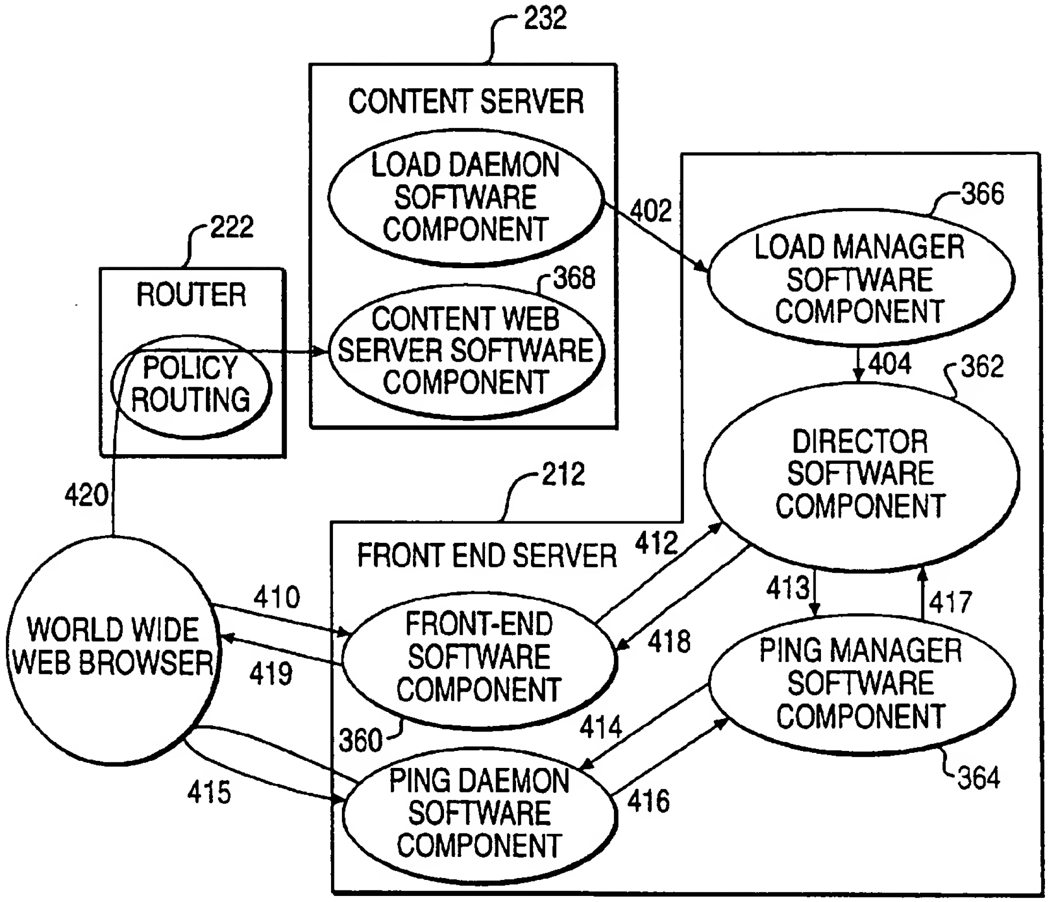

232
CONTENT SERVER
LOAD DAEMON SOFTWARE COMPONENT
CONTENT WEB SERVER SOFTWARE COMPONENT
368
402
222
ROUTER
POLICY ROUTING
366
LOAD MANAGER SOFTWARE COMPONENT
404
362
DIRECTOR SOFTWARE COMPONENT
420
212
FRONT END SERVER
412
413
417
410
WORLD WIDE WEB BROWSER
419
FRONT-END SOFTWARE COMPONENT
418
PING MANAGER SOFTWARE COMPONENT
360
414
364
415
PING DAEMON SOFTWARE COMPONENT
416

FIG. 4B

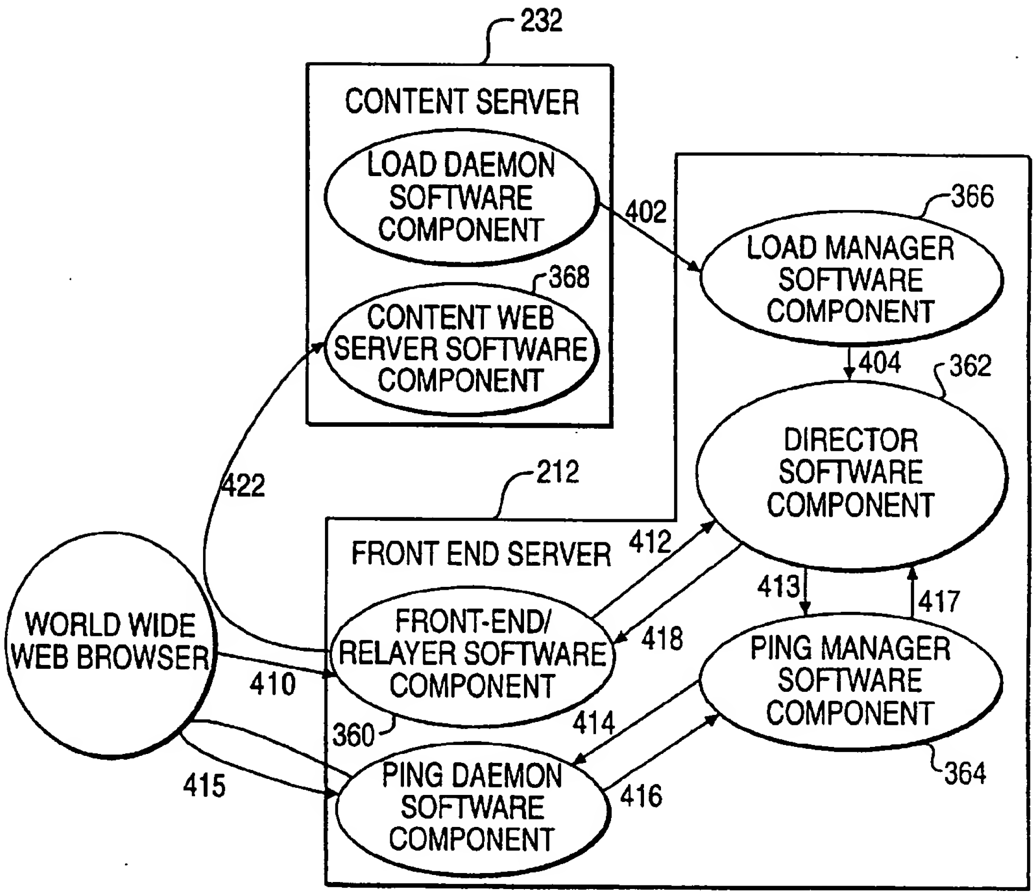

232
CONTENT SERVER
LOAD DAEMON SOFTWARE COMPONENT
368
CONTENT WEB SERVER SOFTWARE COMPONENT
402
366
LOAD MANAGER SOFTWARE COMPONENT
404
362
DIRECTOR SOFTWARE COMPONENT
212
FRONT END SERVER
422
412
418
413
417
WORLD WIDE WEB BROWSER
FRONT-END/ RELAYER SOFTWARE COMPONENT
PING MANAGER SOFTWARE COMPONENT
410
360
414
364
415
PING DAEMON SOFTWARE COMPONENT
416

FIG. 4C

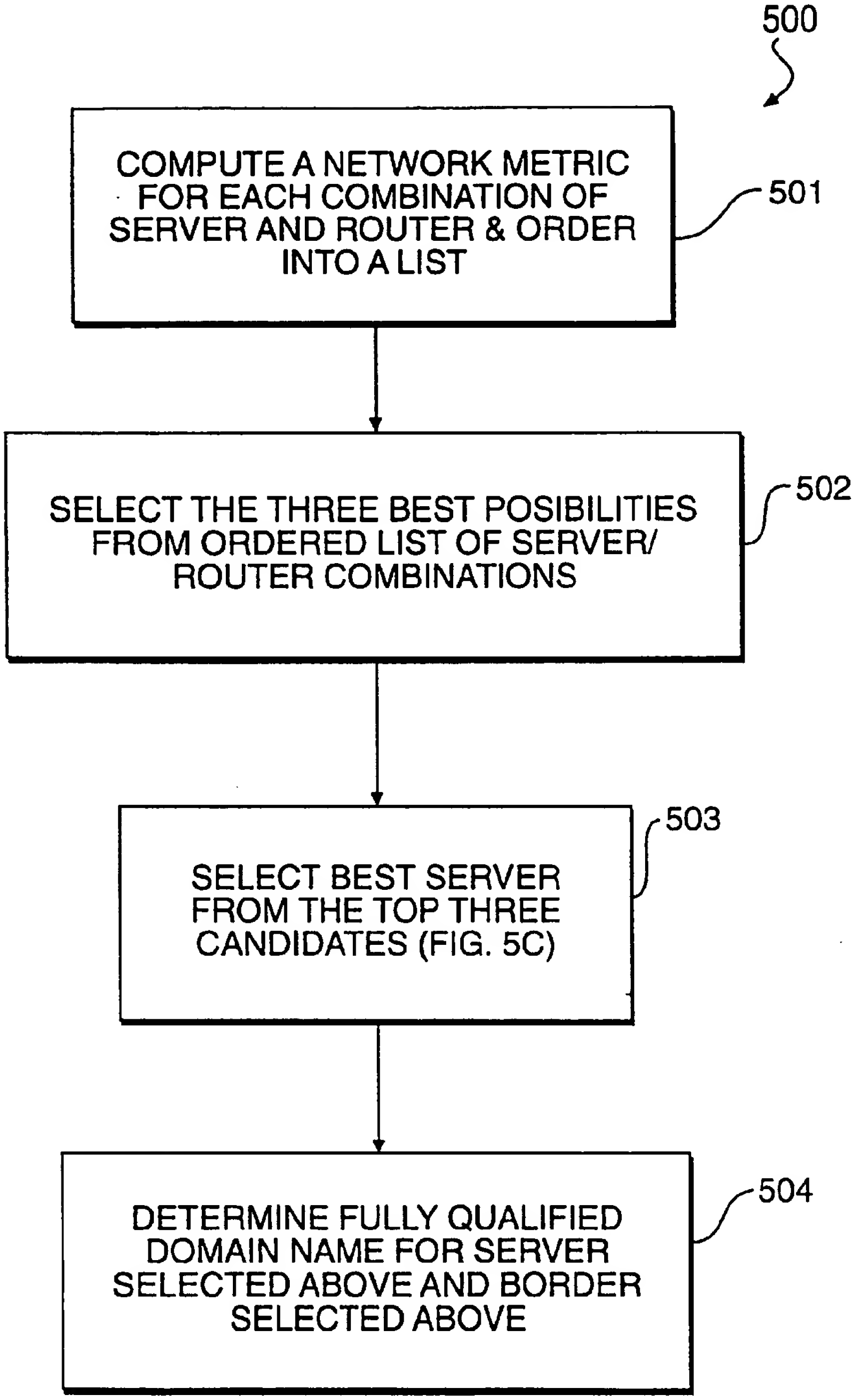

500
COMPUTE A NETWORK METRIC FOR EACH COMBINATION OF SERVER AND ROUTER & ORDER INTO A LIST
501
SELECT THE THREE BEST POSIBILITIES FROM ORDERED LIST OF SERVER/ ROUTER COMBINATIONS
502
SELECT BEST SERVER FROM THE TOP THREE CANDIDATES (FIG. 5C)
503
DETERMINE FULLY QUALIFIED DOMAIN NAME FOR SERVER SELECTED ABOVE AND BORDER SELECTED ABOVE
504

FIG. 5A

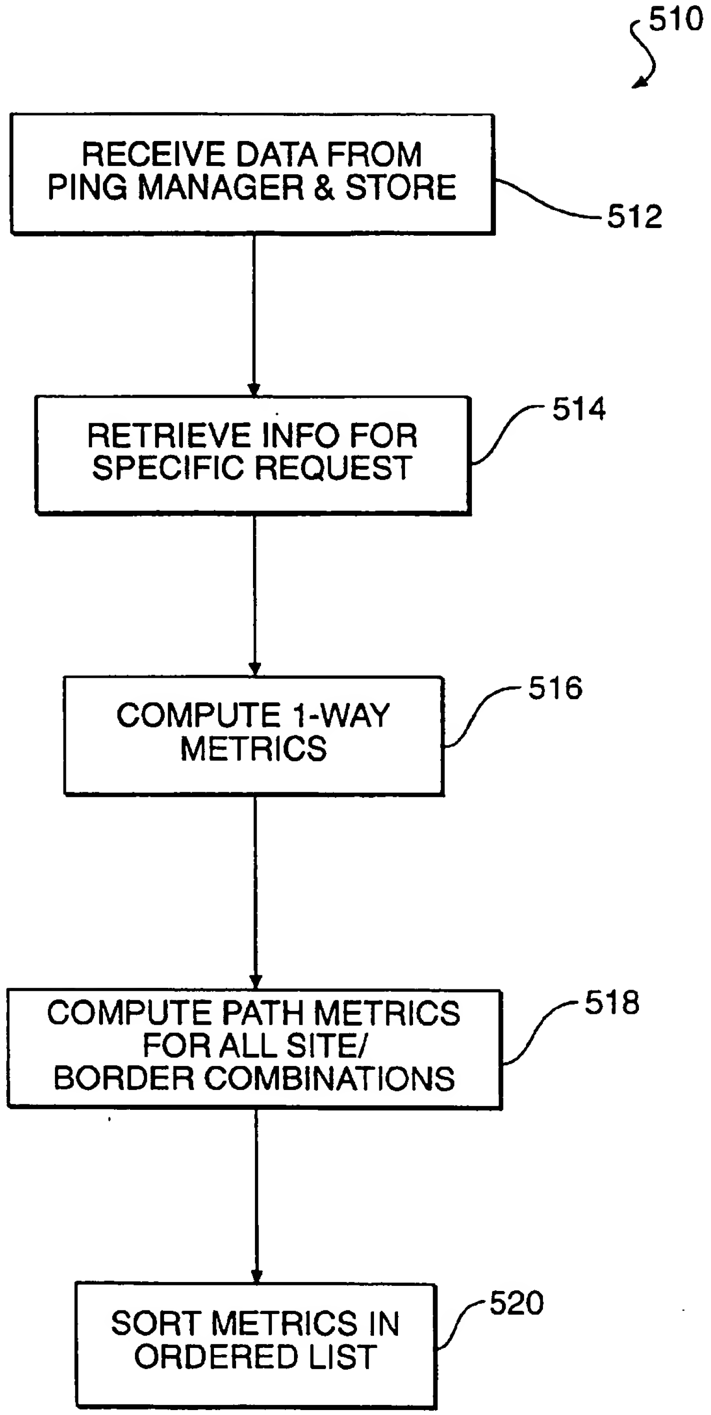

510
RECEIVE DATA FROM PING MANAGER & STORE
512
RETRIEVE INFO FOR SPECIFIC REQUEST
514
COMPUTE 1-WAY METRICS
516
COMPUTE PATH METRICS FOR ALL SITE/ BORDER COMBINATIONS
518
SORT METRICS IN ORDERED LIST
520

FIG. 5B

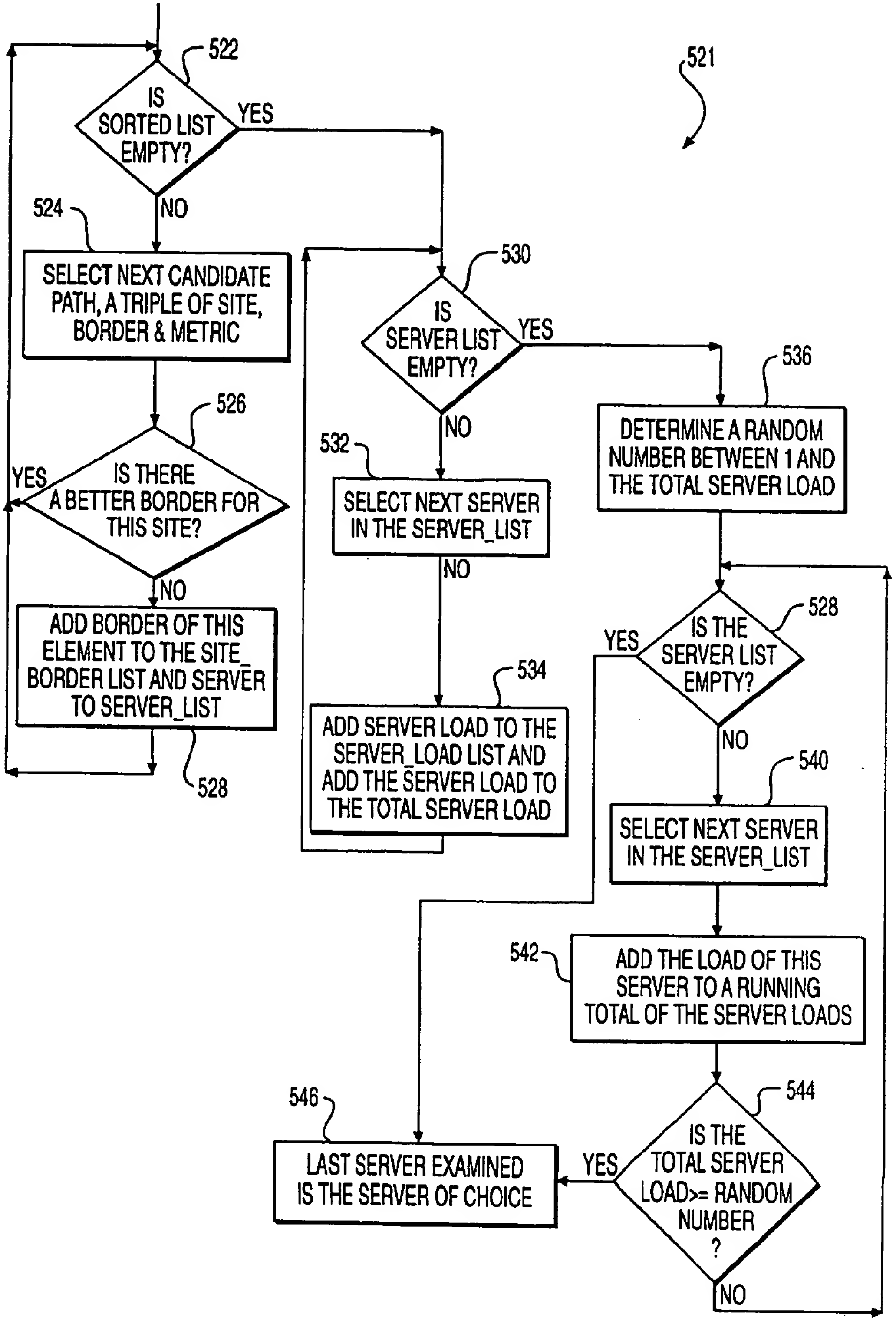

521
522
IS SORTED LIST EMPTY?
YES
NO
524
SELECT NEXT CANDIDATE PATH, A TRIPLE OF SITE, BORDER & METRIC
526
IS THERE A BETTER BORDER FOR THIS SITE?
YES
NO
ADD BORDER OF THIS ELEMENT TO THE SITE_ BORDER LIST AND SERVER TO SERVER_LIST
528
530
IS SERVER LIST EMPTY?
YES
NO
532
SELECT NEXT SERVER IN THE SERVER_LIST
NO
534
ADD SERVER LOAD TO THE SERVER_LOAD LIST AND ADD THE SERVER LOAD TO THE TOTAL SERVER LOAD
536
DETERMINE A RANDOM NUMBER BETWEEN 1 AND THE TOTAL SERVER LOAD
528
IS THE SERVER LIST EMPTY?
YES
NO
540
SELECT NEXT SERVER IN THE SERVER_LIST
542
ADD THE LOAD OF THIS SERVER TO A RUNNING TOTAL OF THE SERVER LOADS
544
IS THE TOTAL SERVER LOAD>= RANDOM NUMBER ?
YES
NO
546
LAST SERVER EXAMINED IS THE SERVER OF CHOICE

FIG. 5C

METHOD AND APPARATUS FOR BALANCING THE PROCESS LOAD ON NETWORK SERVERS ACCORDING TO NETWORK AND SERVE BASED POLICIES

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority from the following U.S. Provisional Application, the disclosure of which, including all appendices and all attached documents, is incorporated by reference in its entirety for all purposes:

U.S. Provisional patent application Ser. No. 60/032,484, Rodney L. Joffe, et. al., entitled, "A Distributed Computing System and Method for Distributing User Requests to Replicated Network Servers", filed Dec. 9, 1996.

COPYRIGHT NOTICE

BACKGROUND OF THE INVENTION

The present invention relates generally to the field of distributed computer systems, and more specifically to computing systems and methods for assigning requests to one of a multiplicity of network servers based upon best criteria such as the speed of the underlying network infrastructure.

The explosive growth of the World Wide Web needs little introduction. Not only are members of the technical community finding an ever greater number of technical and informational resources available on the World Wide Web, but also the mainstream populace is finding favorite restaurants, car makes and churches sporting new websites. The popularity of the World Wide Web as a communications medium lies in the richness of its information content and ease of use. Information in this medium exists as objects in a widely distributed collection of internetworked servers, each object uniquely addressable by its own uniform resource locator (URL). Since its inception, the World Wide Web has achieved a global prominence in everyday life and commerce.

Yet this explosive growth has not been had without difficulty. The proliferation of commercial applications brings with it an ever increasing number of users making ever increasing numbers of inquiries. The problems of latency and bandwidth constraints manifest themselves in delay, lost information and the distraught customers.

Network architects respond using an array of solutions. Many responses fall within the category of solutions based upon supplying more computing power. This may encompass such alternatives as different web server software, web server hardware or platform, increases in RAM or CPU in the server, application rewrites, or increasing network bandwidth by upgrading hardware. Another class of solutions involves using multiple servers or locating servers strategically. One method in this class is to locate the server at the internet service provider. By selecting a service provider with a optimal pairing capability, colocating the server at the service provider's site can yield a much better connection to the rest of the internet. Another approach is the use of distributed servers. Place identical content servers at strategic locations around the world. For example, one in New York, one in San Francisco, and one in London. This distributes the load to multiple servers and keeps traffic closer to the requester. Another approach is to cluster servers. Clustering enables sharing of hard drive arrays across multiple servers. Another approach is the server farm. This entails the use of multiple webservers with identical content, or the segmentation based upon functionality. For example, two servers for web functions, two for FTP, two as a database and so forth. A variation on the server farm is the distributed server farm. This places server farms at strategic locations—essentially combining the server farm with the distributed server approach.

The multiple and distributed server approaches solve one problem at the expense of creating another. If there are multiple servers, how does the end user locate your site? Presently, names and universal resource locators (URLs) are resolved into unique single addresses by a domain name service (DNS). DNS servers maintain a list of domain names cross referenced to individual IP addresses. However, if multiple web servers or server farms are used, the DNS system must be modified. A common approach to this problem is to modify the DNS system to a one to many mapping of names to IP addresses. Thus the DNS will return a list of IP addresses for any particular web object. These may then be handed out to the various clients in a round-robin fashion. There are, however, several drawbacks to this approach. The round-robin paradigm returns IP addresses in a strict order with little regard to the location of the requester or the server. The scheme has no knowledge of server architecture or loading. The selection simply progresses down a simple list. One server may receive all heavy duty users. Additionally the weakest link determines the overall performance, so server platforms need to be kept in relatively parity. Another problem is that the DNS simply returns IP addresses with no regard as to whether the server to which the address corresponds is operational. Consequently, if one of the round-robin servers happens to be off-line for maintenance, DNS continues to give out the address, and potential users continue to receive time-out error responses. Thus, the round-robin modification of DNS makes a broad attempt to solve the distributed server problem. However, there is no regard to network traffic, low balancing among servers or reliability issues.

Several products on the market purport to address these problems, but these prior efforts all suffer the drawback that they require that the user software environment be modified in order to facilitate replicated server selection. A scheme that requires user software modifications is less desirable due to the practical problems of ensuring widespread software distribution. Such schemes are, at best, useful as optimization techniques.

One such class of approaches are those that rely on explicit user selection to assign a user request to a server. The user application may include additional steps that require the user to have sufficient knowledge, sophistication, and patience to make their own server selection. Such schemes are not always desirable for a number of reasons.

A technique based on selective host routing uses multiple replicated servers, all with the same network address, located at different points in the network topology. A router associated with each server captures incoming network traffic to the shared server address, and forwards the traffic to the specific server. This technique can only statically distribute the client request load to the nearby server, with no consideration of server load or other network characteristics.

The BIND implementation of the Domain Name System server may include techniques to bind server names to different network addresses, where one of a set of different multiple addresses are assigned either sequentially or randomly. Service providers assign a different address to each replicated server, and BIND directs user requests to the alternative servers. This technique can only statically distribute the client request load to an arbitrary server, with no consideration of server load or other network characteristics.

SONAR is an emerging IETF (Internet Engineering Task Force) protocol for distributing network characteristics, in particular, for topological closeness. SONAR includes a data format for representing query requests and responses, but does not specify a mechanism for determining the network characteristics.

The Cisco Local Director is a product that works as a network traffic multiplexor that sits in front of multiple local servers, and distributes new transport connections to each server based on the amount of traffic flowing to the servers. This product does not consider network characteristics in its decision, and further requires that the replicated servers be collocated. Cisco Systems is a company headquartered in San Jose, Calif.

The Cisco Distributed Director redirects user requests to topologically distant servers based on information obtained from network routing protocols. The Distributed Director intercepts either incoming DNS requests or HTTP requests, and provides the appropriate response for redirection. This product does not consider server load, and only considers the restricted set of information available from routing protocols; this information is also limited in accuracy by the aggregation techniques required to enable scalable internet routing.

Although these products, taken together, consider server load and network characteristics, they do not make an integrated server selection. Yet, for all these efforts, the pundits' criticism still rings true, it is still the "world wide wait." For this reason, what is needed is a system which automatically selects an appropriate server from which to retrieve a data object for a user based upon the user's request, and the capabilities and topology of the underlying network.

SUMMARY OF THE INVENTION

The present invention provides the ability to assign requests for data objects made by clients among multiple network servers. The invention provides a distributed computing system and methods to assign user requests to replicated servers contained by the distributed computing system in a manner that attempts to meet the goals of a particular routing policy. Policies may include minimizing the amount of time for the request to be completed. For example, a system according to the invention may be configured to serve data objects to users according to the shortest available network path.

Specifically, the invention provides a system for routing requests for data objects from any number of clients based upon a "best server" routing policy to one of multiple content servers. Content servers serve data objects responsive to clients' requests via one or more network access points, in accordance with the decision of a director. The director determines based upon the routing policy the routing of said requests for data objects to a particular content server.

In accordance with a particular aspect of the invention, routing policies may comprise any of the following, a combination of any of the following, or none of the following: 1) the least number of open TCP connections; 2) the most available free RAM; 3) the most available free SWAP (virtual memory); 4) the highest amount of CPU idle time; or 5) the fastest ICMP route to the client's machine.

Advantages of the approaches according to the invention are increased tolerance of faults occurring in the underlying hardware and reliability over prior art web servers. The invention will be better understood upon reference to the following detailed description and its accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1A depicts a representative client server relationship in accordance with a particular embodiment of the invention;

FIG. 1B depicts a functional perspective of the representative client server relationship in accordance with a particular embodiment of the invention;

FIG. 1C depicts a representative internetworking environment in accordance with a particular embodiment of the invention;

FIG. 1D depicts a relationship diagram of the layers of the TCP/IP protocol suite;

FIG. 2A depicts a distributed computing environment in accordance with a particular embodiment of the invention;

FIG. 2B depicts a distributed computing environment in accordance with an alternative embodiment of the invention;

FIG. 3A depicts the relationship of processes in accordance with a representative embodiment of the invention;

FIG. 3B depicts the relationship of processes in accordance with an alternative embodiment of the invention;

FIG. 3C depicts the relationship of processes in accordance with a preferable embodiment of the invention;

FIG. 4A depicts process steps in accordance with a particular embodiment of the invention;

FIG. 4B depicts process steps in accordance with an alternative embodiment of the invention;

FIG. 4C depicts process steps in accordance with a preferable embodiment of the invention; and

FIGS. 5A–5C depict flow charts of the optimization process within a director component according to a particular embodiment of the invention.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

1.0 Introduction

A preferable embodiment of a server load balancing system according to the invention has been reduced to practice and will be made available under the trade name "HOPSCOTCHT™."

A word about nomenclature is in order. Systems according to the present invention comprise a multiplicity of processes which may exist in alternative embodiments on any of a multiplicity of computers in a distributed networked environment or as concurrent processes running in virtual machines or address spaces on the same computer. To limit the exponential growth of names, the following conventions have been employed to enhance readability. Individual deviations will be noted where they occur. An 'xyz server' is a computer or virtual machine housing a collection of processes which make up xyz. An 'xyz component' is a collection of processes which perform a set of functions collectively referred to as xyz. An 'xyz' is the set of functions being performed by the xyz component on the xyz machine.

1.1 Hardware Overview

The distributed computing system for server load balancing (the "system") of the present invention is implemented in the Perl programming language and is operational on a computer system such as shown in FIG. 1A. This invention may be implemented in a client-server environment, but a client-server environment is not essential. FIG. 1A shows a conventional client-server computer system which includes a server 20 and numerous clients, one of which is shown as client 25. The use of the term "server" is used in the context of the invention, wherein the server receives queries from (typically remote) clients, does substantially all the processing necessary to formulate responses to the queries, and provides these responses to the clients. However, server 20 may itself act in the capacity of a client when it accesses remote databases located at another node acting as a database server.

The hardware configurations are in general standard and will be described only briefly. In accordance with known practice, server 20 includes one or more processors 30 which communicate with a number of peripheral devices via a bus subsystem 32. These peripheral devices typically include a storage subsystem 35, comprised of memory subsystem 35*a* and file storage subsystem 35*b*, which hold computer programs (e.g., code or instructions) and data, set of user interface input and output devices 37, and an interface to outside networks, which may employ Ethernet, Token Ring, ATM, IEEE 802.3, ITU X.25, Serial Link Internet Protocol (SLIP) or the public switched telephone network. This interface is shown schematically as a "Network Interface" block 40. It is coupled to corresponding interface devices in client computers via a network connection 45.

Client 25 has the same general configuration, although typically with less storage and processing capability. Thus, while the client computer could be a terminal or a low-end personal computer, the server computer is generally a high-end workstation or mainframe, such as a SUN SPARC™ server. Corresponding elements and subsystems in the client computer are shown with corresponding, but primed, reference numerals.

The user interface input devices typically includes a keyboard and may further include a pointing device and a scanner. The pointing device may be an indirect pointing device such as a mouse, trackball, touchpad, or graphics tablet, or a direct pointing device such as a touchscreen incorporated into the display. Other types of user interface input devices, such as voice recognition systems, are also possible.

The user interface output devices typically include a printer and a display subsystem, which includes a display controller and a display device coupled to the controller. The display device may be a cathode ray tube (CRT), a flat-panel device such as a liquid crystal display (LCD), or a projection device. Display controller provides control signals to the display device and normally includes a display memory for storing the pixels that appear on the display device. The display subsystem may also provide non-visual display such as audio output.

The memory subsystem typically includes a number of memories including a main random access memory (RAM) for storage of instructions and data during program execution and a read only memory (ROM) in which fixed instructions are stored. In the case of Macintosh-compatible personal computers the ROM would include portions of the operating system; in the case of IBM-compatible personal computers, this would include the BIOS (basic input/output system).

The file storage subsystem provides persistent (non-volatile) storage for program and data files, and typically includes at least one hard disk drive and at least one floppy disk drive (with associated removable media). There may also be other devices such as a CD-ROM drive and optical drives (all with their associate removable media). Additionally, the computer system may include drives of the type with removable media cartridges. The removable media cartridges may, for example be hard disk cartridges, such as those marketed by Syquest and others, and flexible disk cartridges, such as those marketed by Iomega. One or more of the drives may be located at a remote location, such as in a server on a local area network or at a site of the Internet's World Wide Web.

In this context, the term "bus subsystem" is used generically so as to include any mechanism for letting the various components and subsystems communicate with each other as intended. With the exception of the input devices and the display, the other components need not be at the same physical location. Thus, for example, portions of the file storage system could be connected via various local-area or wide-area network media, including telephone lines. Similarly, the input devices and display need not be at the same location as the processor, although it is anticipated that the present invention will most often be implemented in the context of PCs and workstations.

Bus subsystem 32 is shown schematically as a single bus, but a typical system has a number of buses such as a local bus and one or more expansion buses (e.g., ADB, SCSI, ISA, EISA, MCA, NuBus, or PCI), as well as serial and parallel ports. Network connections are usually established through a device such as a network adapter on one of these expansion buses or a modem on a serial port. The client computer may be a desktop system or a portable system.

The user interacts with the system using interface devices 37' (or devices 37 in a standalone system). For example, client queries are entered via a keyboard, communicated to client processor 30', and thence to network interface 40' over bus subsystem 32'. The query is then communicated to server 20 via network connection 45. Similarly, results of the query are communicated from the server to the client via network connection 45 for output on one of devices 37' (say a display or a printer), or may be stored on storage subsystem 35'.

FIG. 1B is a functional diagram of the computer system of FIG. 1A. FIG. 1B depicts a server 20, and a representative client 25 of a multiplicity of clients which may interact with the server 20 via the internet 45 or any other communications method. Blocks to the right of the server are indicative of the processing components and functions which occur in the server's program and data storage indicated by block 35*a* in FIG. 1A. A TCP/IP "stack" 44 works in conjunction with Operating System 42 to communicate with processes over a network or serial connection attaching Server 20 to internet 45. Web server software 46 executes concurrently and cooperatively with other processes in server 20 to make data objects 50 and 51 available to requesting clients. A Common Gateway Interface (CGI) script 55 enables information from user clients to be acted upon by web server 46, or other processes within server 20. Responses to client queries may be returned to the clients in the form of a Hypertext Markup Language (HTML) document outputs which are then communicated via internet 45 back to the user.

Client 25 in FIG. 1B possesses software implementing functional processes operatively disposed in its program and data storage as indicated by block 35*a*' in FIG. 1A. TCP/IP stack 44', works in conjunction with Operating System 42' to communicate with processes over a network or serial connection attaching Client 25 to internet 45. Software implementing the function of a web browser 46' executes concurrently and cooperatively with other processes in client 25 to make requests of server 20 for data objects 50 and 51. The user of the client may interact via the web browser 46' to make such queries of the server 20 via internet 45 and to view responses from the server 20 via internet 45 on the web browser 46'.

1.2 Network Overview

FIG. 1C is illustrative of the internetworking of a plurality of clients such as client 25 of FIGS. 1A and 1B and a multiplicity of servers such as server 20 of FIGS. 1A and 1B as described herein above. In FIG. 1C, a network 60 is an example of a Token Ring or frame oriented network. Network 60 links a host 61, such as an IBM RS6000 RISC workstation, which may be running the AIX operating system, to a host 62, which is a personal computer, which may be running Windows 95, IBM OS/2 or a DOS operating system, and a host 63, which may be an IBM AS/400 computer, which may be running the OS/400 operating system. Network 60 is internetworked to a network 70 via a system gateway which is depicted here as router 75, but which may also be a gateway having a firewall or a network bridge. Network 70 is an example of an Ethernet network that interconnects a host 71, which is a SPARC workstation, which may be running SUNOS operating system with a host 72, which may be a Digital Equipment VAX6000 computer which may be running the VMS operating system.

Router 75 is a network access point (NAP) of network 70 and network 60. Router 75 employs a Token Ring adapter and Ethernet adapter. This enables router 75 to interface with the two heterogeneous networks. Router 75 is also aware of the Internetwork Protocols, such as ICMP ARP and RIP, which are described below.

FIG. 1D is illustrative of the constituents of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite. The base layer of the TCP/IP protocol suite is the physical layer 80, which defines the mechanical, electrical, functional and procedural standards for the physical transmission of data over communications media, such as, for example, the network connection 45 of FIG. 1A. The physical layer may comprise electrical, mechanical or functional standards such as whether a network is packet switching or frame-switching; or whether a network is based on a Carrier Sense Multiple Access/Collision Detection (CSMA/CD) or a frame relay paradigm.

Overlying the physical layer is the data link layer 82. The data link layer provides the function and protocols to transfer data between network resources and to detect errors that may occur at the physical layer. Operating modes at the datalink layer comprise such standardized network topologies as IEEE 802.3 Ethernet, IEEE 802.5 Token Ring, ITU X.25, or serial (SLIP) protocols.

Network layer protocols 84 overlay the datalink layer and provide the means for establishing connections between networks. The standards of network layer protocols provide operational control procedures for internetworking communications and routing information through multiple heterogenous networks. Examples of network layer protocols are the Internet Protocol (IP) and the Internet Control Message Protocol (ICMP). The Address Resolution Protocol (ARP) is used to correlate an Internet address and a Media Access Address (MAC) of a particular host. The Routing Information Protocol (RIP) is a dynamic routing protocol for passing routing information between hosts on networks. The Internet Control Message Protocol (ICMP) is an internal protocol for passing control messages between hosts on various networks. ICMP messages provide feedback about events in the network environment or can help determine if a path exists to a particular host in the network environment. The latter is called a "Ping". The Internet Protocol (IP) provides the basic mechanism for routing packets of information in the Internet. IP is a non-reliable communication protocol. It provides a "best efforts" delivery service and does not commit network resources to a particular transaction, nor does it perform retransmissions or give acknowledgments.

The transport layer protocols 86 provide end-to-end transport services across multiple heterogenous networks. The User Datagram Protocol (UDP) provides a connectionless, datagram oriented service which provides a non-reliable delivery mechanism for streams of information. The Transmission Control Protocol (TCP) provides a reliable session-based service for delivery of sequenced packets of information across the Internet. TCP provides a connection oriented reliable mechanism for information delivery.

The session, or application layer 88 provides a list of network applications and utilities, a few of which are illustrated here. For example, File Transfer Protocol (FTP) is a standard TCP/IP protocol for transferring files from one machine to another. FTP clients establish sessions through TCP connections with FTP servers in order to obtain files. Telnet is a standard TCP/IP protocol for remote terminal connection. A Telnet client acts as a terminal emulator and establishes a connection using TCP as the transport mechanism with a Telnet server. The Simple Network Management Protocol (SNMP) is a standard for managing TCP/IP networks. SNMP tasks, called "agents", monitor network status parameters and transmit these status parameters to SNMP tasks called "managers." Managers track the status of associated networks. A Remote Procedure Call (RPC) is a programming interface which enables programs to invoke remote functions on server machines. The Hypertext Transfer Protocol (HTTP) facilitates the transfer of data objects across networks via a system of uniform resource indicators (URI).

The Hypertext Transfer Protocol is a simple protocol built on top of Transmission Control Protocol (TCP). The HTTP provides a method for users to obtain data objects from various hosts acting as servers on the Internet. User requests for data objects are made by means of an HTTP GET request. A GET request as depicted below comprises 1) an HTTP header of the format "http://"; followed by 2) an identifier of the server on which the data object resides; followed by 3) the full path of the data object; followed by 4) the name of the data object. In the GET request shown below, a request is being made of the server "www.w3.org" for the data object with a path name of "/pub/" and a name of "MyData.html":

```
GET http://www.w3.org/pub/MyData.html    (1)
```

Processing of a GET request entails the establishing of a TCP/IP connection with the server named in the GET request and receipt from the server of the data object specified. After receiving and interpreting a request message, a server responds in the form of an HTTP RESPONSE message.

Response messages begin with a status line comprising a protocol version followed by a numeric Status Code and an associated textual Reason Phrase. These elements are separated by space characters. The format of a status line is depicted in line (2):

Status-Line=HTTP-Version Status-Code Reason-Phrase (2)

The status line always begins with a protocol version and status code, e.g., "HTTP/1.0 200". The status code element is a three digit integer result code of the attempt to understand and satisfy a prior request message. The reason phrase is intended to give a short textual description of the status code. The first digit of the status code defines the class of response. There are five categories for the first digit. 1XX is an information response. It is not currently used. 2XX is a successful response, the action was successfully received, understood and accepted. 3XX is a redirection response indicating that further action must be taken in order to complete the request. It is this response that is used by certain embodiments of the present invention to cause a client to redirect to a selected server site. 4XX is a client error response. This indicates a bad syntax in the request. Finally, 5XX is a server error. This indicates that the server failed to fulfill an apparently valid request.

Particular formats of HTTP messages are described in, Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, UDEL, August 1982, which is incorporated by reference herein for all purposes.

2.0 Specific Configurations

FIG. 2A depicts a representative distributed computing system according to the present invention. In FIG. 2A, a network **200** interconnects a plurality of server machines with one another and to an external internetworking environment via a plurality of Network Access Points (NAPs). The topology of this internal network is completely arbitrary with respect to the invention. It may be Ethernet, Token Ring, Asynchronous Transfer Mode (ATM) or any other convenient network topology. The Network Access Points are points of connection between large networks, and may comprise routers, gateways, bridges or other methods of joining networks in accordance with particular network topologies.

Network access points **202, 204, 206** and **208** provide network reachability to external networks A, B, C and D for communication with client machines via a plurality of external network paths. In a particular embodiment, these Network Access Points house a portion of the routing configuration component. In a preferable embodiment, these NAPs are routers **222, 224, 226** and **228** that peer with other networks and are Open Shortest Path First (OSPF) routing algorithm aware. OSPF is capable of accomodating the case where several machines have exactly the same IP address, by routing packets to the closest machine. By contrast an alternative routing mechanism, Routing Information Protocol (RIP), would not handle this case.

Further, these routers employ IP tunneling techniques, as are well known to persons of ordinary skill in the art, and are policy routing capable, i.e., able to route packets based in part on their source address. Each is configured to use IP tunneling to route packets out of the network through a particular router at a particular NAP, based on packet source addresses and server availability.

FIG. 2A also shows Front End Servers **212, 214, 216** and **218**, that are co-located at each NAP. These Front End Servers house a Front End component process. In a preferable embodiment, the Front End Servers also house an IP Relayer component process. Functions of these processes are described hereinbelow.

A Director server **250** houses several software components, which include in the preferable embodiment a Director component, a Ping Manager component, and a Load Manager component, each of which will be described hereinbelow.

One or more Content Servers **232, 234, 236** and **238** perform the actual serving of content, i.e., Web pages or FTP files. Each has associated with it one or more instances of a Content server component capable of serving data such that outgoing packets have a source address that is selectable by the Director component.

Each Content Server has a network alias IP address for each router controlled by the system. For example, in a system with four routers, a particular content server will have four separate network alias IP addresses. Routing within the system is configured for "policy routing", i.e., routing based on the source address of packets, so that depending on which network alias IP address the packets are from, they will be routed through a specific system router chosen by the Director. A series of selectable IP tunnels enable the Content Servers to send data to clients using a "best route". IP tunnels are configured such that each server can serve data out through a chosen network access point. By contrast, systems commonly known in the art serve data routed out of the network through a default peering point, i.e., network exit points that can route to the other networks which make up the Internet. Typically the default peering point is the only peering point for the network. Each IP tunnel is configured to send all data to a different peer router. Every IP tunnel starts at one router, and ends at another (remote) router. This enables a content server farm (one or more content servers in the same physical location, serving behind a router) to serve data through a different content server farm's router if the Director has determined this to be the optimal path.

Routers are statically configured to send packets down an IP tunnel when the packets have a source address that is associated with that IP tunnel. For example, in a network having three peering points the router at a first peering point, call it "A", would have two outgoing tunnels, "tunnel 1" and "tunnel 2". These outgoing tunnels lead to the other two routers, each residing at one of the other two peering points, B and C. The router at peering point A would be configured so that it routes all packets that have a source address of 1.1.1.1 down tunnel **1** and all packets that have a source address of 2.2.2.2 down tunnel 2. Tunnel **1** sends all packets to a second peering point, B, and tunnel 2 sends all packets to a third peering point, C. This is source-based routing, commonly known as "Policy Routing".

A Content Server associated with the router at peering point A runs server software that binds the local side of its server sockets to addresses 1.1.1.1 and 2.2.2.2, enabling it to serve from either address. The Director software possesses configuration information about this Content Server and its server addresses. This enables the Director to determine that when the Content Server serves from the 1.1.1.1 address, all packets are served via tunnel 1 to network access point B, and when the Content Server serves from the 2.2.2.2 address, all packets are served via tunnel **2** to network access point C. The Director software is able to select the NAP through which each Content Server will serve its data by informing the client which fully qualified domain name, corresponding to an IP address, to access for service, since that will be the source address of the request's reply packets. The reply packets are automatically routed through the NAP chosen by the Director software.

This policy routing configuration is set up once for each system during installation. The Director has access to a table of IP addresses for each content server, and the corresponding system router for each of the IP addresses of a particular content server. This enables the Director to select a Content Server IP address that will route through the system router of the Director's choice. The Director must first decide which Content Server has the least load. It formulates this from the data given to it by the Load Manager (which, in turn, collected the data from each of the Load Daemons). Once the Director has chosen the least-loaded Content Server machine, it will choose an address from that machine's set of network alias IP addresses that routes through a router having the best ICMP one-way trip time to the browsing client. It makes this decision based on data given to it by the Ping Manager (which in turn collected its data from the Ping Daemons).

FIG. 2B depicts an alternative embodiment of a distributed computing system according to the present invention. The embodiment of FIG. 2B differs from the embodiment of FIG. 2A, primarily in that the embodiment of FIG. 2B does not have a separate Director server. Rather, in the alternative embodiment of FIG. 2B, the processes which resided on the Director server **250** in FIG. 2A are distributed among the front end servers **212**, **214**, **216** and **218**, in FIG. 2B.

3.0 Specific Processes

FIG. 3A depicts the process components of a representative embodiment according to the present invention.

3.1 Process Components

The Front End—An embodiment of a front-end component **360** receives client requests for data objects. This may be an incoming HTTP request in the preferable embodiment. The front end next solicits "advice" from the director components **362**, if available, by immediately sending the browsing client's IP address to the Director, and waiting for the Director to select the "best" server. Finally, it sends a reply to the requesting client that directs the client to contact a specific server for further request processing. The front-end must understand the protocol of the client request, and will use application-specific mechanisms to direct the client to the specific server. In a particular embodiment, the front end sends the browser client an HTTP redirection response to the best server's URL. An embodiment of this invention may comprise multiple front-end components that understand one or more user-level protocols.

The Director—An embodiment of a director component **362** receives data queries from front-end components **360** and, using data about the source of the client, preferably, the browsing client's IP address, as well as replicated server status and network path characteristics, preferably ICMP echo response times, received from the collector components, such as the Ping Manager **364** and the Load Manager **366**, returns information that enables front-ends to direct user requests. The decision takes all data into account and sends the front end the IP address of the "best" server. Director components include the decision methods to evaluate alternative server selections, and provide coordination for the entire system. An embodiment of this invention comprises one or more director components.

The Collector Components—An embodiment of a collector component monitors one or more characteristics of network paths or content server load, and make this information available to the director component for use in the server selection decision. An embodiment of this invention comprises one or more collector components, preferably, a Ping Manager Component **364**, a Ping Daemon Component (not shown), a Load Manager Component **366** and a Load Daemon Component (not shown).

The Ping Manager tells the Director which content server has the fastest ICMP echo path. This data is collected by the Ping Manager **364**. The Ping Manager receives its ping-time data from individual server machines which are each using ICMP pings to determine the ICMP routing time between themselves and the browsing client's machine. The Ping Manager then stores this information and reports it to the Director, which uses it to make decisions about the best route.

The Ping Daemon executes on server machines associated with each content server machine cluster. A content server machine (or cluster of them) resides near each of the Network Access Points. The Ping Daemon waits for a ping request (and its corresponding IP address, which is the browsing client's IP address) and then pings the browsing client's IP address to record the ICMP routing time through its own closest border router. It then sends this data back to the Ping Manager.

The Load Manager software is similar to the Ping Manager, but reports and stores information from the Load Daemon about each of the content server machines' current load. It forwards this data to the Director as well.

The Load Daemon runs in conjunction with each content server **368** and reports back to the Load Manager periodically. It sends data about the number of currently open TCP connections, free RAM, free SWAP, and CPU idle time.

FIG. 3B depicts the software components of an alternative embodiment according to the present invention. Comparing software component diagram of alternative embodiment of FIG. 3B with that of the embodiment in FIG. 3A, the main difference between the two embodiments is that in the alternative embodiment depicted by FIG. 3B, the Director process **362** is distributed among various servers. Thus, in FIG. 3B the Director process **363** is shown as three separate instances of a director process. Whereas, in FIG. 3A the Director process **362** is shown as a singular Director process with interfaces to other processes on multiple servers.

FIG. 3C depicts the software components of a preferable embodiment according to the present invention. Comparing the software component diagram of the embodiment of FIG. 3B with that of the embodiment in FIG. 3C, the main difference between the two embodiments is that in the embodiment depicted by FIG. 3C, the Front End process **360** includes an IP Relayer function.

The Front End/IP Relayer—An embodiment of a front-end/IP Relayer component **360** receives incomming IP packets for any IP traffic. As in the embodiments of FIGS. 3A and 3B, the front end next solicits "advice" from the director components **362**, if available, by immediately sending the client's IP address to the Director, and waiting for the Director to select the "best" server. However, rather than directing the client to contact a specific server for further request processing, as is performed by the Front End components in the embodiments of FIGS. 3A and 3B, the IP Relayer forwards the packets to the chosen "best server" in accordance with the determination made by the Director. An embodiment of this invention may comprise multiple front-end components functioning at the IP layer.

3.2 Steps to Service a Request

FIG. 4A depicts a set of steps that occur in the process of receiving, evaluating, and answering a client's request in a particular embodiment of the invention. In a step **402**, a load daemon process resident on a content server, such as **232** of FIG. 2A, is periodically updating information directly to a load manager process **366** residing on a Director server **250**. Subsequently, in a step **404**, Load Manager process updates load information gathered from all content servers having load daemon processes. This enables the Director process **362** to choose the least loaded content server machine responsive to an incoming request such as in steps **410** and **412**.

In step **410**, an incoming client request, which may be an HTTP request from for example a Web browser, or an FTP request from an FTP client, is routed via an arbitrary external path to a system border gateway, typically a router at a Network Access Point. At this point the client's request becomes an input to the server load balancing distributed computing system of the present invention. It is forwarded to a front-end component **360** by IP acting in accordance with the routing table associated with a router located at this NAP. In step **412**, the front-end component, responsive to the arrival of the client application's request, makes a request of the Director component **362** for a preferred server. The Director, having received a front-end request for information about client request in step **412**, requests information, such as the most expedient path between servers and clients, from a collector component, such as a Ping Manager **364**, in a step **413**. In the embodiment of FIG. 4A, the most expedient path is the path with the fastest ICMP echo reply (Ping), determined by a Ping Manager Collector Component **364** acting in conjunction with one or more Ping Daemon Components co-located at the various NAPs in the system network in a step **414**. These Ping Daemons determine the fastest ICMP echo path between their particular front end server and the client by transmitting successive Pings to the client via their particular front end server's associated NAP, and timing the response, as depicted in a step **415**. Each Ping Daemon from time to time transmits the time to a client via its particular NAP to the Ping Manager in a step **416**. The Ping Manager Component returns round trip values for paths associated with the system's NAPs to the client in a step **417**. In specific embodiments, the Ping Manager Component may initiate pro-active status queries (not depicted), or may return status information at any time, without explicit requests from the Director component. The Director component indicates the correct content server to be used for the client to the front end component in a step **418**. The front-end component directs the client to the correct content server using an application layer protocol, preferably an HTTP redirect response in a step **419**. The front-end response is forwarded via a NAP to the client machine by the internetwork using, for example, the IP protocol. Subsequently, as shown in step **420**, requests from the client will be made to the best content server, e.g., **232**, via the best route to his machine. (Which is also the best place to enter the external network to get to the browsing client's own network provider.)

FIG. 4B depicts a set of steps that occur in the process of receiving, evaluating, and answering a client's request in an alternative embodiment of the invention. Comparing the processing steps of the embodiment depicted in FIG. 4B with those of the embodiment depicted in FIG. 4A, it is clear that the particular steps are identical. However, it is noteworthy that steps **412**, **414**, **416** and **418** are not network transactions, as they were in FIG. 4A, but rather transactions between co-resident processes within one server machine.

FIG. 4C depicts a set of steps that occur in the process of receiving, evaluating, and answering a client's request in a preferable embodiment of the invention. Comparing the processing steps of the embodiment depicted in FIG. 4B with those of the embodiment depicted in FIG. 4C, it is clear that the main difference is that steps **419** and **420** of FIG. 4B, the redirect response and the subsequent HTTP conversation steps, respectively, have been replaced by a new step **422**. In step **422**, the IP traffic from the client is relayed to the "best server" in accordance with the determination of the Director. This takes the place of the redirect response step **419** being made by the Front End process to the client as is shown in FIG. 4B. It is noteworthy that steps **402**, **404**, **410**, **412**, **413**, **414**, **415**, **416**, **417** and **418** remain the same in the preferable embodiment of FIG. 4C as in the embodiments of FIGS. **4A** and **4B**.

FIG. 4C depicts the processing steps in the preferable embodiment. In a step **410**, a Front-End/Relayer **360** intercepts packets entering the network. Based upon address information contained within the IP header, the Front-End/Relayer determines the original destination server of the packet. Next, in a step **412**, the Front-End/Relayer calls upon the Director **362** to make the "best server" routing decision. In steps **413**, **414**, **414**, **416**, **417** and **418**, the Director, in conjunction with the Load Manager **366**, Ping Manager **364**, Load Daemon and Ping Daemon components, determines a "best server" for the IP traffic and communicates this machine's address to the Front-End/Relayer. The decision process is identical to that of the embodiments of FIGS. **4A** and **4B**. In a step **422**, the Front-End/Relayer relays all packets from that client to the "best" server machine. Since the packet relaying takes place at the IP level, packets for any service running over the IP layer can be routed by the methods of the present invention.

4.0 Decision-Making Methodologies

4.1 Director

The Director makes decisions about which Content Server and which router is best for each request by a Client. FIG. **5A** depicts a flowchart **500** of the process steps undertaken by a Director in a specific embodiment of the invention. In a step **501**, a network metric is calculated for each combination of "content server" and "outgoing router". A sorted list is constructed of these metrics, having the format:

(site1 border1 metric1, . . . , siteN borderN metricN)   (3)

In a step **502**, the candidate servers for the best sites are selected from the list produced in step **501**, above. Processing traverses the list, selecting the top ranking metric, as well as any candidates that are within a certain percentage (say X %) of the top ranking metric. Note that each content server will be listed with all combinations of outgoing routers in the list. Since it is not necessary to consider server-router combinations which are nonsensical, for example from the LA server going through the NY border need not be considered if there is a combination of LA server going through the LA border, i.e, only consider the "best." For example, with X=5, and the sorted list of combinations (site, border, metric) as is depicted in lines (4):

@(network=(ny ny 300 sj la 305 sj sj 312 ny sj 380 dc ny 400 . . .)   (4)

The first three entries are:

ny ny 300
sj la 305
sj sj 312

Since these are all within 5% of each other, the system will consider all three. However, note that San Jose is listed twice, and that the time of the second listing is longer than the time of the first. The first appearance of San Jose server therefore, is preferred over the second, thus the second is discarded in favor of the first. A server in either NY or SJ is selected. If a server in NY is chosen, then the border in NY will also be selected. If a server in SJ is chosen, the border in LA is "best."

In a step **503**, the "best" server is selected from the candidates. Make a list of all of the metrics for candidate servers, and apply a statistical algorithm to select the best server from the candidates. The result of this step is the site for the selected server, and an identifier for the specific server. (There may be multiple servers at each site).

In a step **504**, from the site of the server selected above, the system recalls which border is to be used for that site saved in step **502**. Then, from the server identifier and the outgoing border, the correct fully qualified domain name that has the appropriate IP address for the internal policy routing is determined.

4.2 Ping Manager

The Ping Manager constantly requests ping information from one or more Ping Daemons. The Ping Manager sends to the Director a sequence of values representative of the round trip time to-and-from every client site. Non-responsive client sites are represented by an arbitrary large value if a Ping Daemon does not respond. In a particular embodiment a Ping Manager will send to a Director a sequence such as depicted by lines (5) and (6) below:

```
"client_address metric_site_1 metric_site2 . . . metric_site_N
  \n"                                                          (5)

"128.9.192.44 300 9999999 280 450 \n"                          (6)
```

In a preferred embodiment the site metrics are ordered.

```
@incoming_metric_order=('ny', 'la', 'dc', 'sj')                (7)
```

FIG. **5B** depicts a flowchart **510** of the process steps performed by the Director in response to receiving from the Ping Manager the sequence of information depicted in lines (5) and (6) above. In a step **512** of flowchart **510**, the Director performs processing on the incoming information from the Ping Manager to store the information in a usable format as depicted by the pseudocode in lines (8) herein below.

```
($addr, @metrics)=split(", $incoming_pingmgr_message);      (8)

$ping_cache{$addr}=$incoming_ping_message;
$ping_cache_time{$addr}=&get_current_time0;
# Store round-trip ping metrics in useful format.
#
@temp_keys=@incoming_metric_order;
while ($value=shift(@metrics)) {$key=shift(@temp_
  keys);
$ping{$key}=$value;
}
```

Next, in a step **514**, in response to a request received from a Front End process, the Director will retrieve information about that request to be answered. Lines (9) depict pseudocode for processing step **514**:

```
Retrieve info about the request that must be answered.      (9)

#
$request_frontend=$request_frontend{$addr}; # who gets
  reply
$contact_site=$request_contact {$addr}; # need for path
  calcs
```

In a step **516**, the Director calculates one-way metrics from the two-way metrics provided by the Ping Manager according to the following pseudocode in lines (10):

```
# Calculate one-way metrics.                                 (10)

#
$plway{$contact_site}=$ping{$contact_site}/ 2;
foreach $border ('ny', 'la', 'dc', 'sj') {next if ($border eq
  $contact_site);
$p1way{$border}=$ping{$border}
  $pl way {$contact_site}
  $internal{"$contact_site $border"};
}
```

Next, in a step **518**, the Director calculates path metrics for all site and border combinations according to the following pseudocode in lines (11):

```
## Calculate the path metrics for all site/border combinations. (11)

#
foreach $site ('ny', 'la', 'dc', 'sj') {foreach $border ('ny',
  'la', 'dc', 'sj') {
  # This calculates round-trip from server, through border,
  # to client, back through 1 st contact border, to server.
  # It assumes that $ping{} array has one-way metric times.
  # Also use $internal{} associate array.
  #
# We weigh outgoing path components twice as much,
  # because we believe it is more important to consider
  # outgoing data path.
#
$metric=2* $p1way{$border}+
  2 * $internal{"$site $border"}+
  $p1way{$contact_site}+
  $internal {"$border $contact_site"};
# save it for easy sorting
  #
$path{$metric}=($path{$metric})?
  "$path{$metric} $site $border": "$site $border"; }
}
```

Next, in a step **520**, the Director sorts the metrics and constructs an ordered list using the following pseudocode in lines (12):

```
# Sort metrics and construct required list in order.        (12)

#
foreach $metric (sort keys %/path) {
  @list=split("$path{$metric});
while (@list){
  $site,=shift(@list);
  $border=shift(@list);
  push(@sorted_list, $site, $border, $metric);
  }
}
```

At this point, variable @sorted_list contains all the information needed to select pairs of sites and candidate servers.

4.3 Load Manager

The Load Manager sends messages to the Director about once every two seconds, in the format depicted in line (13):

```
"server1 load1 server2 load2 . . . serverN loadN";          (13)
```

FIG. **5C** depicts a flowchart **521** of the process steps performed by the Director in selecting a "best" server, using the information received from the Load Manager, depicted in line (13) above, and the Ping Manger, depicted in lines (5) and (6) above.

The Director maintains several internal data structures, including an associative array of server loads, pairs load values with server identifiers:

```
$load_array{$serverID}=$load;
```

a list of servers at each site:

```
$servers {'la'}="www.la1.test.com www.la2.test.com";
$servers {'dc'}="www.dc1.test.com www.dc2.test.com";
```

a mapping of servers and borders into a correct name that has a corresponding source address:

```
$server_map{"www.la1.test.com 1a"}="www.la1-
   la.test.com";
$server_map{"www.la1.test.com dc"}="www.la1-
   dc.test.com";
```

The variable @sorted_list, generated above, contains triples of sites, borders, and path metrics. From these data structures, the Director can choose an optimal server using the steps shown in FIG. 5C. A decisional step verifies that all entries in the @sorted_list have not been processed, and initializes the "best_metric" variable to the metric of the first member of the @sorted_list. These steps are depicted in the following pseudocode:

```
%site_border=0;
@server_list=0;
## Top loop chooses next set of candidate paths (and
   sites);
## loop is broken when we have selected a server.
#
while (1) {
break unless (@sorted_list);
$best_metric=$sorted_list[2] * $fuzz_factor; # $fuzz_
   factor=1.05;
```

As shown in FIG. 5C, a decisional step **522**, a process step **524**, a decisional step **526**, and a process step **528** implement a looping construct for selecting triples of site, border and metric from the @sorted_ list and adding the border information to a site border list and the site to a server list. This is also depicted in the following pseudocode:

```
while (@sorted_list && ($sorted_list[2]<=$best_
   metric)) {
$site=shift(@sorted_list);
$border=shift(@sorted_list);
$metric=shift(@sorted_list);
# check if site already has better outgoing border
if (! $site_border{$site}) {$site_border{$site}=$border;
   push(@server_list, split('', $servers{$site}));
}
}
```

Once this loop has processed all members of the @sorted_ list, decisional step **522** will take the "yes" path, and processing will continue with a decisional step **530**, which, along with process steps **532** and **534**, implements a looping construct for processing all servers in the server list generated above, and adding each server's load to a server load list, and totaling the loads of all servers in a $total variable. This is also depicted in the following pseudocode:

```
@server_load=0;
$total=0;
foreach $server (@server_list) {push(@server_load,
   $load_array{$server}); $total+=$load_
   array{$server};
}
next unless ($total); # check for all servers busy
```

Once this loop has processed all members of the @server_ list, decisional step **530** of FIG. 5C will take the "yes" path, and processing will continue with a process step **536**, which determines a random number between 1 and the total server load. Next, processing continues in the loop formed by decisional steps **538** and **544** and processing steps **540**, **542**, and **546**. This loop steps through the servers in the @server_list, summing their loads until this running total exceeds the random number selected in process step **536**, or until the last member of the @server_list has been processed. In either case, the server being examined when either of these two conditions is fulfilled, is the server selected by the Director as the "best" server. This is also depicted in the following pseudocode:

```
# tricky to handle fencepost values.
# Should return value between 1 and $total (inclusive);
#
$random=roll_dice($total);
$total=0;
while ($the_server=shift(@server_list)) {$value=shift
   (@server_load);
next unless ($value);
$total +=$value;
last if ($random<=$total);
}
last; ## We have answer, break out of while loop . . .
}
```

Once the director has selected a "best" server, processing continues as described by step **504** of FIG. 5A, and the following pseudo code:

```
# Selected server is at which site???
#
$site=$get_site{$the_server};
# Recall information about best border for this site
#
$border=$site_border{$site};
# Get the correct name which will use source address we
   need.
#
$response_to_request=$server_map{"$the_server
   $border"};
```

5.0 Conclusion

In conclusion, it can be seen that the present invention provides for an internetworked system wherein data objects are served to users according to the shortest available network path. A further advantage to the approaches according to the invention is that these methods exhibit tolerance of faults occurring in the underlying hardware. Additionally, reliability of systems according to the invention are increased over prior art web servers. Other embodiments of the present invention and its individual components will become readily apparent to those skilled in the art from the foregoing detailed description, wherein is described embodiments of the invention by way of illustrating the best mode contemplated for carrying out the invention. As will be realized, the invention is capable of other and different embodiments and its several details are capable of modifications in various obvious respects, all without departing from the spirit and the scope of the present invention. Accordingly, the drawings and detailed description are to be regarded as illustrative in nature and not as restrictive. It is therefore not intended that the invention be limited except as indicated by the appended claims.

What is claimed is:

1. A system for routing requests for data objects from a plurality of clients comprising:

a plurality of content servers configured to serve said data objects;

a director component for routing said requests for data objects, said director component further comprising:

a server selection component configured to select a content server from said plurality of content servers based upon a policy;

an identification component configured to identify a plurality of routes from a requesting client to said content server;

a determination component configured to determine a time to traverse each of said plurality of routes; and

a route selection component configured to:

select a route from said requesting client to said content server having the shortest time to traverse; and

identify one of a plurality of IP addresses associated with said content server that directs content to the requesting client, from the content server, along the selected route.

2. The system of claim 1 wherein said director component includes

an answer component for answering a request for a data object with a response that redirects a request from one of said plurality of clients to said selected content server along said route.

3. The system of claim 2 wherein said answer component includes a redirection component that uses an HTTP redirect response.

4. The system of claim 1 wherein said determination component includes a measurement component that measures the time to traverse said route using an ICMP echo reply.

5. The system of claim 1 wherein said server selection component includes:

a determination component configured to determine the number of open TCP connections for each content server in said plurality of content servers; and

a selection component configured to select a content server having the least number of open TCP connections.

6. The system of claim 1 wherein said server selection component includes:

a determination component configured to determine the amount of available free RAM for each content server in said plurality of content servers; and

a selection component configured to select a content server having the largest amount of available free RAM.

7. The system of claim 1 wherein said server selection component includes:

a determination component configured to determine the amount of available free SWAP for each content server in said plurality of content servers; and

a selection component configured to select a content server having the largest amount of available free SWAP.

8. The system of claim 1 wherein said server selection component includes:

a determination component configured to determine the amount of CPU idle time for each content server in said plurality of content servers; and

a selection component capable of selecting the content server having the highest amount of CPU idle time.

9. A method for routing requests for data objects from a plurality of clients comprising the steps of:

receiving a request for a data object from one of said plurality of clients;

providing a plurality of content servers capable of serving data objects, wherein each of said plurality of content servers are comprised of a plurality of IP addresses that each route data objects from said content servers along a different path;

determining a best server from said plurality of content servers according to a policy;

identifying a plurality of routes from said client to said best server;

determining a time to traverse each of said plurality of routes;

selecting one of said routes having the shortest time to traverse; and

informing said client of one of said plurality of IP addresses associated with the best server that directs content to the client, from the best server, along said route.

10. The method of claim 9 wherein said informing a particular client further comprises:

answering a request for a data object with a response that causes said particular client to redirect said request to said best server.

11. The method of claim 10 wherein said response that redirects uses an HTTP redirect response.

12. The method of claim 9 wherein determining said best server according to said policy further comprises selecting a least loaded content server from said plurality of content servers.

13. The method of claim 12 wherein determining said best server further comprises:

determining the number of open TCP connections for each content server in said plurality of content servers; and

selecting as the least loaded content server the content server having the least number of open TCP connections.

14. The method of claim 12 wherein determining a best server further comprises:

determining the amount of available free RAM for each content server in said plurality of content servers; and

selecting as the least loaded content server the content server having the largest amount of available free RAM.

15. The method of claim 12 wherein determining said best server further comprises:

determining the amount of available free SWAP for each content server in said plurality of content servers; and

selecting as the least loaded content server the content server having the largest amount of available free SWAP.

16. The method of claim 12 wherein determining said best server further comprises:

determining the amount of CPU idle time for each content server in said plurality of content servers; and

selecting as the least loaded content server the content server having the highest amount of CPU idle time.

17. The method of claim 9 wherein said time to traverse is determined by measuring the time associated with an ICMP echo reply made over said route.

18. A method for routing information from a plurality of clients comprising the steps of:

receiving a request for a data object from one of a plurality of clients;

determining a best server from said plurality of content servers according to a policy associated with a packet networking environment;

determining one of a plurality of routes from said requesting client to said best server, said route having the shortest time to traverse; and

relaying said request back to said client, wherein said request is comprised of one of a plurality of IP addresses associated with said best server that directs content corresponding to the clients request to the client along said route.

19. The method of claim 18 wherein said relaying information from a particular client further comprises:

forwarding said information to said best server using network layer protocols.

20. The method of claim 18 wherein determining a best server further comprises selecting a least loaded content server from said plurality of content servers.

21. The method of claim 18 wherein said determining a route further comprises:

identifying a plurality of routes from said requesting client to said best server;

determining a time to traverse each of said plurality of routes; and

selecting one of said plurality of routes having the shortest time to traverse said route.

22. The method of claim 21 wherein said time to traverse is determined by an ICMP echo reply.

23. The method of claim 18 wherein determining a best server further comprises:

determining the number of open TCP connections for each content server in said plurality of content servers; and

selecting as the best server the content server having the least number of open TCP connections.

24. The method of claim 18 wherein determining a best server further comprises:

determining the amount of available free RAM for each content server in said plurality of content servers; and

selecting as the best server the content server having the largest amount of available free RAM.

25. The method of claim 18 wherein determining a best server further comprises:

determining the amount of available free SWAP for each content server in said plurality of content servers; and

selecting as the best server the content server having the largest amount of available free SWAP.

26. The method of claim 18 wherein determining a best server further comprises:

determining the amount of CPU idle time for each content server in said plurality of content servers; and

selecting as the best server the content server having the highest amount of CPU idle time.

27. A system for routing requests for data objects from a plurality of clients comprising:

plurality of content servers that serve said data objects, each of said plurality of content servers comprising a plurality of IP addresses; and

a director component that identifies a least loaded server based upon a policy that:

selects said least loaded server from one of said plurality of servers;

selects one of a plurality of routes from a requesting client to said least loaded server, wherein said route has the shortest time to traverse; and

identifies an IP address associated with the least loaded server that directs content to the client along said route.

28. The system of claim 27 wherein said director component further includes:

a selection component capable of selecting a server from said plurality of servers based upon said policy; and

an answer component capable of answering a request for a data object by indicating to one of said plurality of clients to redirect the request to said server selected by said selection component.

29. A method of routing requests for data objects from a plurality of clients

comprising the steps of:

providing a plurality of content servers that serve said data objects; and

routing said requests for data objects back to a requesting client for subsequent transmission to one of said plurality of servers, wherein each of said requests is comprised of one of a plurality of IP addresses associated with said content server that directs content corresponding to the client's request from said content server along a route having the shortest time to traverse.

* * * * *

US006449647B1

(12) **United States Patent**
Colby et al.

(10) **Patent No.: US 6,449,647 B1**
(45) **Date of Patent: *Sep. 10, 2002**

(54) **CONTENT-AWARE SWITCHING OF NETWORK PACKETS**

(75) Inventors: **Steven Colby**, Billerica; **John J. Krawczyk**, Arlington; **Rai Krishnan Nair**, Acton, all of MA (US); **Katherine Royce**, Manchester; **Kenneth P. Siegel**, Nashua, both of NH (US); **Richard C. Stevens**, Littleton; **Scott Wasson**, Shrewsbury, both of MA (US)

(73) Assignee: **Cisco Systems, Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/400,635**

(22) Filed: **Sep. 21, 1999**

Related U.S. Application Data

(63) Continuation of application No. 09/050,524, filed on Mar. 30, 1998, now Pat. No. 6,006,264.

(60) Provisional application No. 60/054,687, filed on Aug. 1, 1997.

(51) **Int. Cl.**^{7} **G06F 15/173**; G06F 15/177

(52) **U.S. Cl.** **709/226**; 709/220; 709/240

(58) **Field of Search** 709/226, 239, 709/240, 245, 224, 250

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,644,532	A		2/1987	George et al.	370/94
5,031,089	A		7/1991	Liu et al.	364/200
5,230,065	A		7/1993	Curley et al.	395/200
5,249,290	A		9/1993	Heizer	395/650
5,341,477	A		8/1994	Pitkin et al.	395/200
5,371,852	A		12/1994	Attanasio et al.	395/200
5,459,837	A		10/1995	Caccavale	395/184.01
5,475,685	A		12/1995	Garris et al.	370/82
5,475,819	A		12/1995	Miller et al.	395/200.03
5,574,861	A		11/1996	Lorvig et al.	395/200.06
5,581,703	A		12/1996	Baugher et al.	395/200.06
5,603,029	A	*	2/1997	Aman et al.	395/675
5,673,393	A		9/1997	Marshall et al.	395/200.04
5,701,465	A		12/1997	Baugher et al.	395/610
5,913,040	A	*	6/1999	Rakavy et al.	709/232
5,974,460	A		10/1999	Maddalozzo, Jr. et al.	709/224
5,983,281	A		11/1999	Ogle et al.	709/249
6,067,559	A	*	5/2000	Allard et al.	709/202
6,138,162	A	*	10/2000	Pistriotto et al.	709/229
6,141,759	A	*	10/2000	Braddy	713/201

OTHER PUBLICATIONS

Resonate, Inc.—Products—Datasheets, http://www.resonate.com/products/intro.html, downloaded May 26, 1998, publication date 1997.

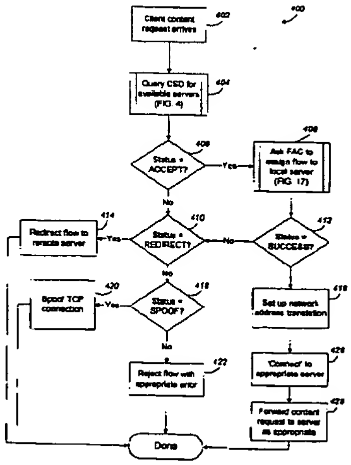
(List continued on next page.)

Primary Examiner—Zarni Maung
(74) *Attorney, Agent, or Firm*—Chapin & Huang, L.L.C.; Barry W. Chapin

(57) **ABSTRACT**

A content-aware flow switch intercepts a client content request in an IP network, and transparently directs the content request to a best-fit server. The best-fit server is chosen based on the type of content requested, the quality of service requirements implied by the content request, the degree of load on available servers, network congestion information, and the proximity of the client to available servers. The flow switch detects client-server flows based on the arrival of TCP SYNs and/or HTTP GETs from the client. The flow switch implicitly deduces the quality of service requirements of a flow based on the content of the flow. The flow switch also provides the functionality of multiple physical web servers on a single web server in a way that is transparent to the client, through the use of virtual web hosts and flow pipes.

56 Claims, 26 Drawing Sheets


400
Client content request arrives
402
Query CSD for available servers (FIG. 4)
404
Status = ACCEPT?
406
Yes
Ask FAC to assign flow to local server (FIG. 17)
408
No
Status = REDIRECT?
410
Redirect flow to remote server
414
Yes
No
Status = SUCCESS?
412
No
Status = SPOOF?
418
Spoof TCP connection
420
Yes
Set up network address translation
416
No
Reject flow with appropriate error
422
"Connect" to appropriate server
424
Forward content request to server as appropriate
426
Done

OTHER PUBLICATIONS

Nair et al., "Robust Flow Control for Legacy Data Appplication over Integrated Services ATM Networks", Global Information Infrastructure (Gil) Evolution, IOS Press, pp. 312–321, 1996.

Sedgewick, Algorithms in C, pp. 353–357, Addison–Wesley Publishing Company, Oct. 1997.

Joffe et al., Hòpscotch White Paper, http://www.genuity.net/products–services/hopscotch–wp.html, downloaded Nov. 4, 1997, publication date unknown.

HydraWEB Technologies, http://www.hydraweb.com/, downloaded Jul. 30, 1997, publication date unknown.

Comer, Internetworking with TCP/IP 1:163–167, publication date unknown.

* cited by examiner

FIG. 1a

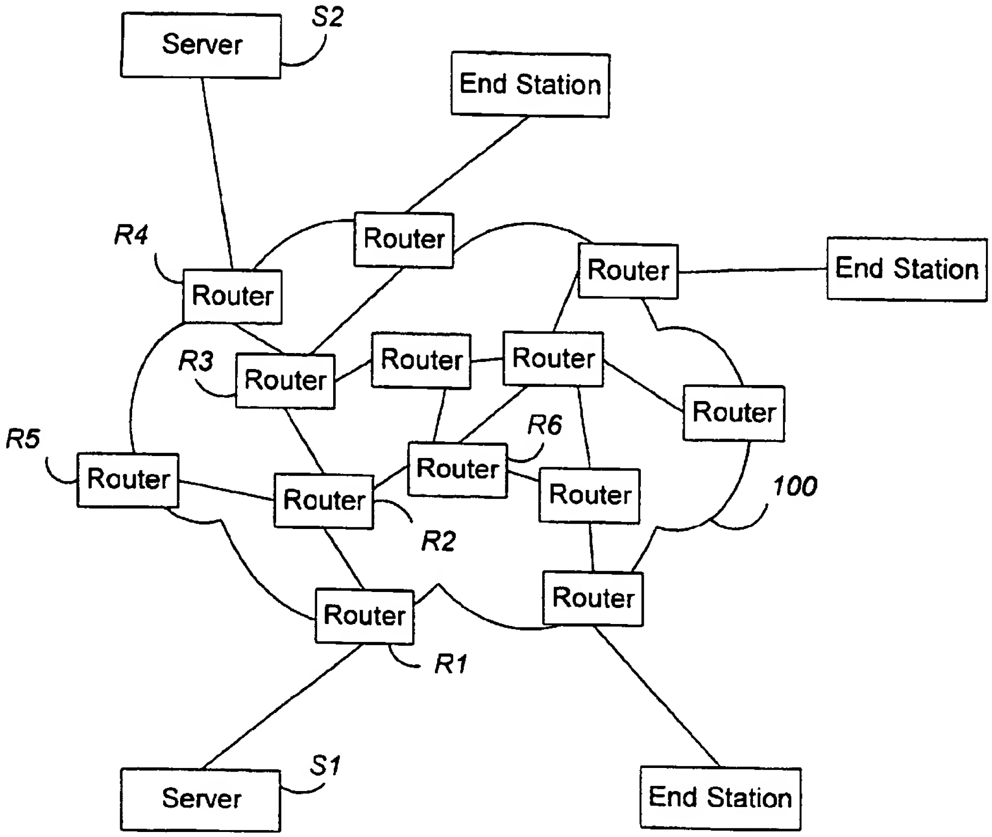

Server
S2
End Station
Router
R4
Router
Router
End Station
R3
Router
Router
Router
Router
R6
R5
Router
Router
Router
Router
100
R2
Router
Router
R1
S1
Server
End Station

FIG. 1b

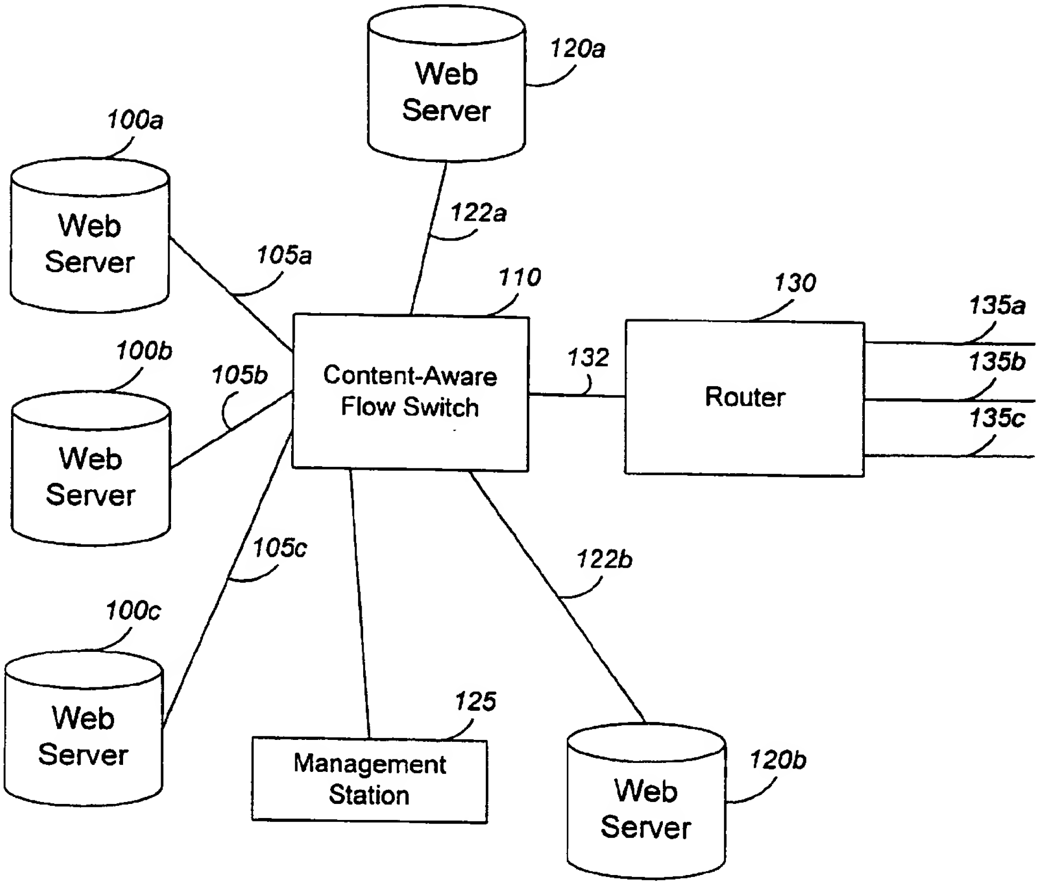

120a
Web Server
100a
Web Server
122a
105a
110
130
135a
100b
105b
Content-Aware Flow Switch
132
Router
135b
Web Server
135c
105c
122b
100c
Web Server
125
Management Station
120b
Web Server

FIG. 1c

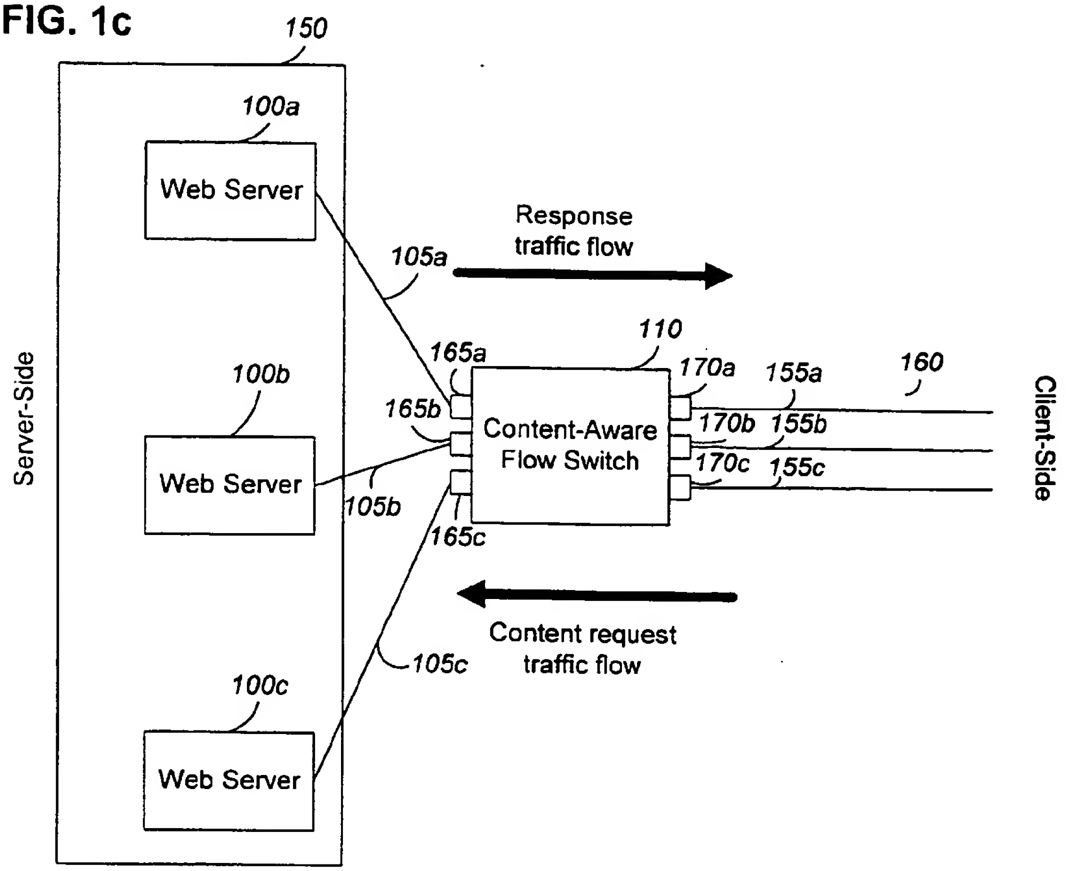

150
100a
Web Server
100b
Web Server
100c
Web Server
Server-Side
105a
105b
105c
Response traffic flow
Content request traffic flow
110
165a
165b
165c
Content-Aware Flow Switch
170a
170b
170c
155a
155b
155c
160
Client-Side

FIG. 2

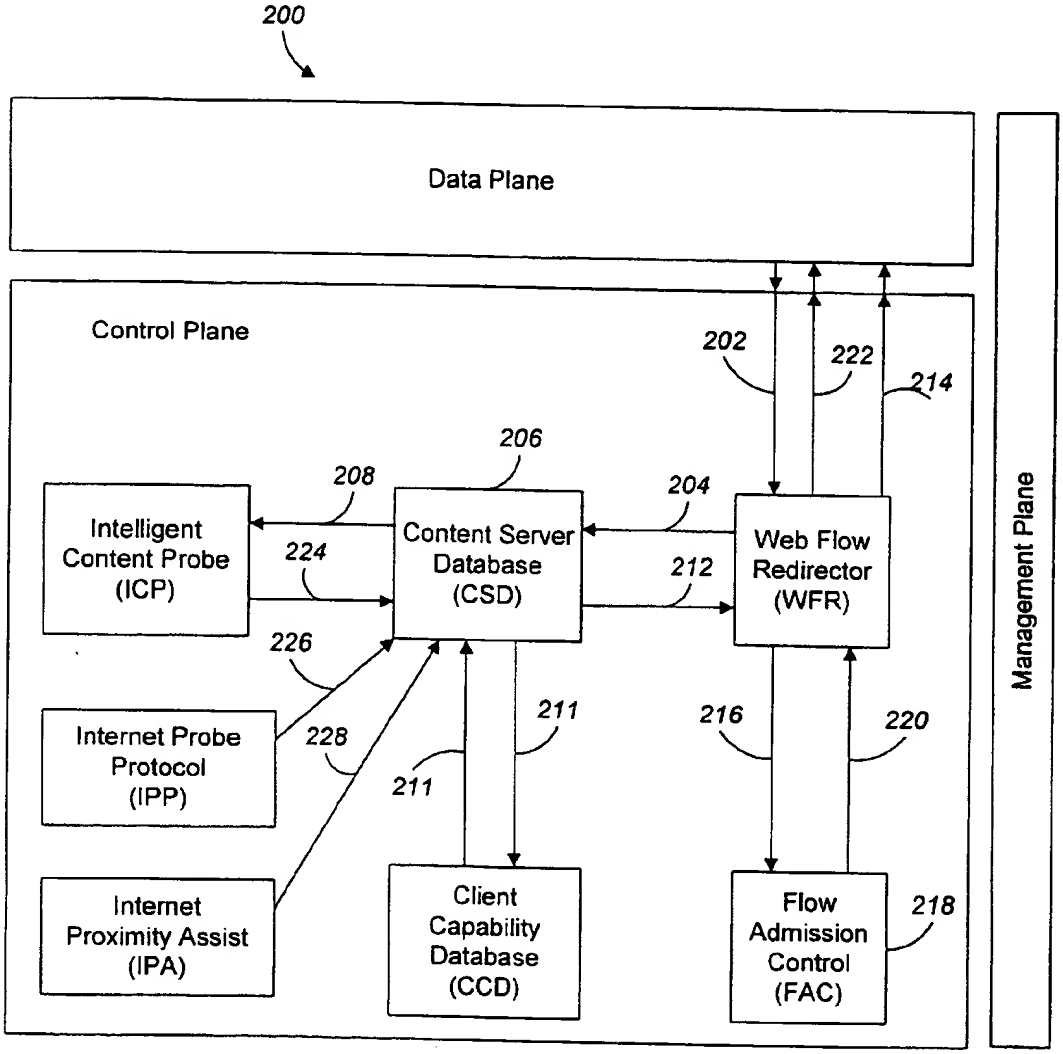

200
Data Plane
Control Plane
Management Plane
202
222
214
206
208
204
Intelligent Content Probe (ICP)
224
Content Server Database (CSD)
212
Web Flow Redirector (WFR)
226
Internet Probe Protocol (IPP)
228
211
211
216
220
Internet Proximity Assist (IPA)
Client Capability Database (CCD)
Flow Admission Control (FAC)
218

FIG. 3

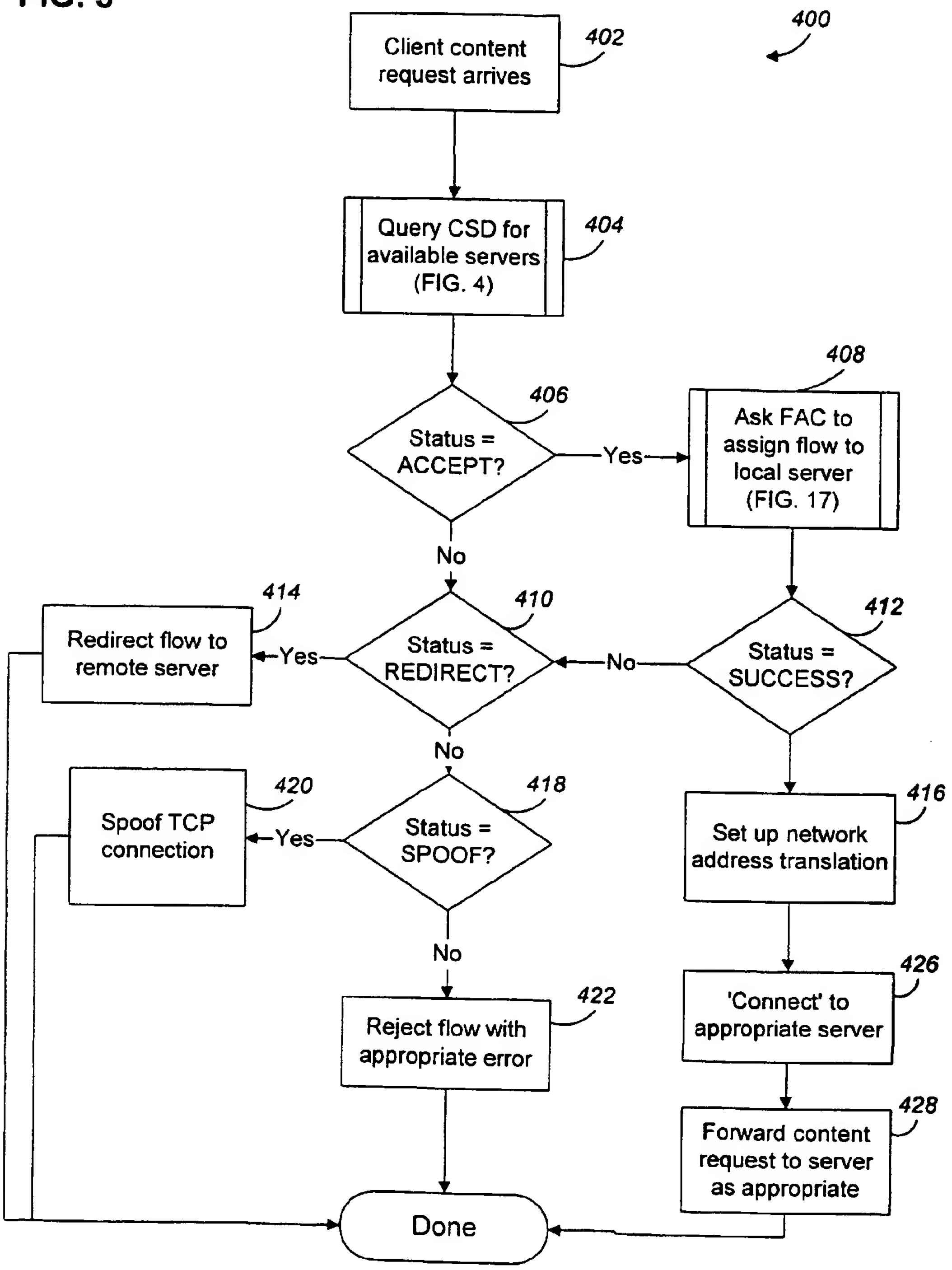

400
Client content request arrives
402
Query CSD for available servers (FIG. 4)
404
Status = ACCEPT?
406
Yes
Ask FAC to assign flow to local server (FIG. 17)
408
No
Status = REDIRECT?
410
Yes
Redirect flow to remote server
414
No
Status = SUCCESS?
412
No
Status = SPOOF?
418
Yes
Spoof TCP connection
420
No
Reject flow with appropriate error
422
Set up network address translation
416
'Connect' to appropriate server
426
Forward content request to server as appropriate
428
Done

FIG. 4

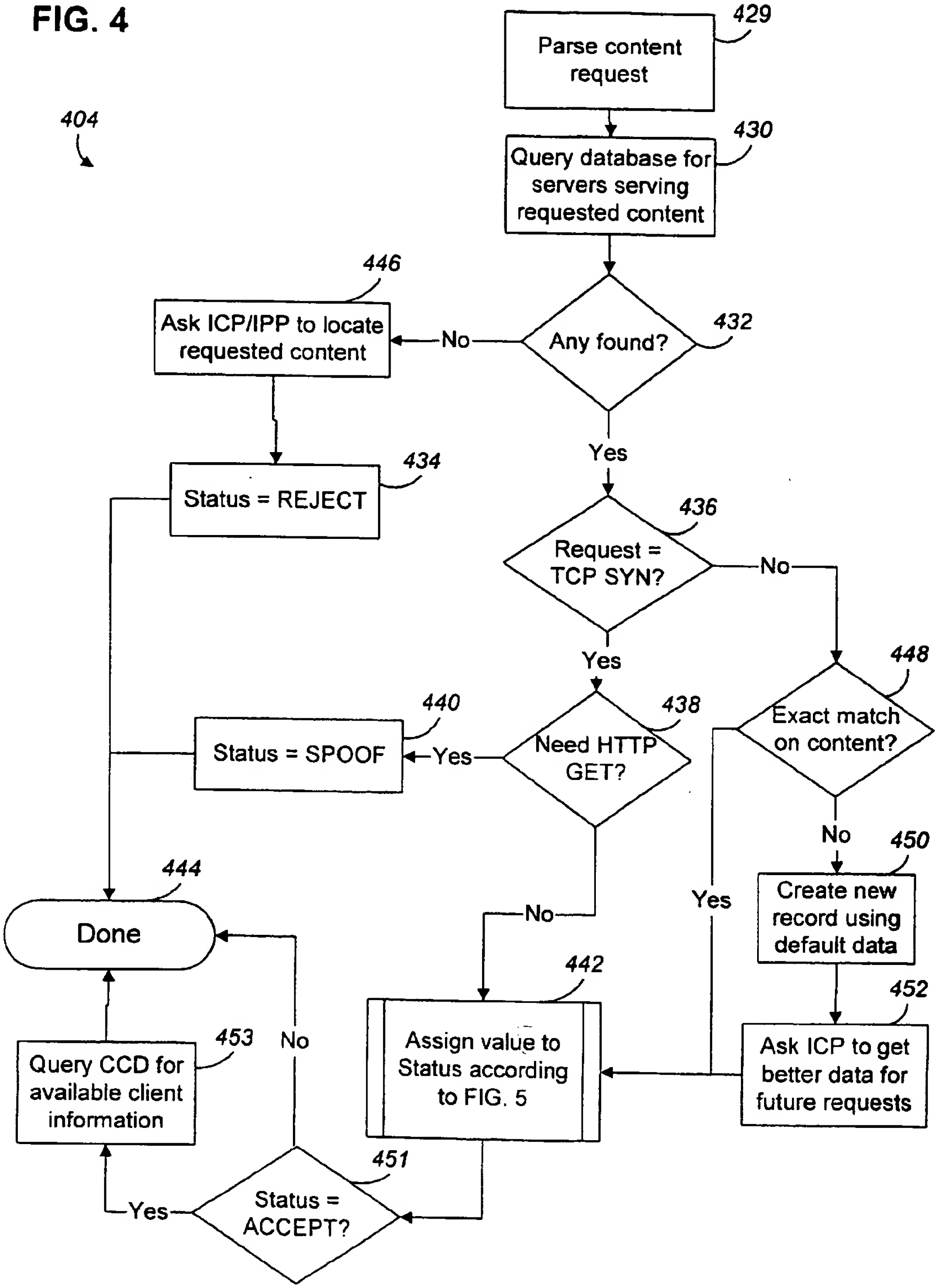

404
Parse content request
429
Query database for servers serving requested content
430
Any found?
432
No
Ask ICP/IPP to locate requested content
446
Yes
Status = REJECT
434
Request = TCP SYN?
436
No
Yes
Exact match on content?
448
Need HTTP GET?
438
Yes
Status = SPOOF
440
No
450
Create new record using default data
Yes
Done
444
No
Assign value to Status according to FIG. 5
442
Ask ICP to get better data for future requests
452
Query CCD for available client information
453
No
Status = ACCEPT?
451
Yes

FIG. 5

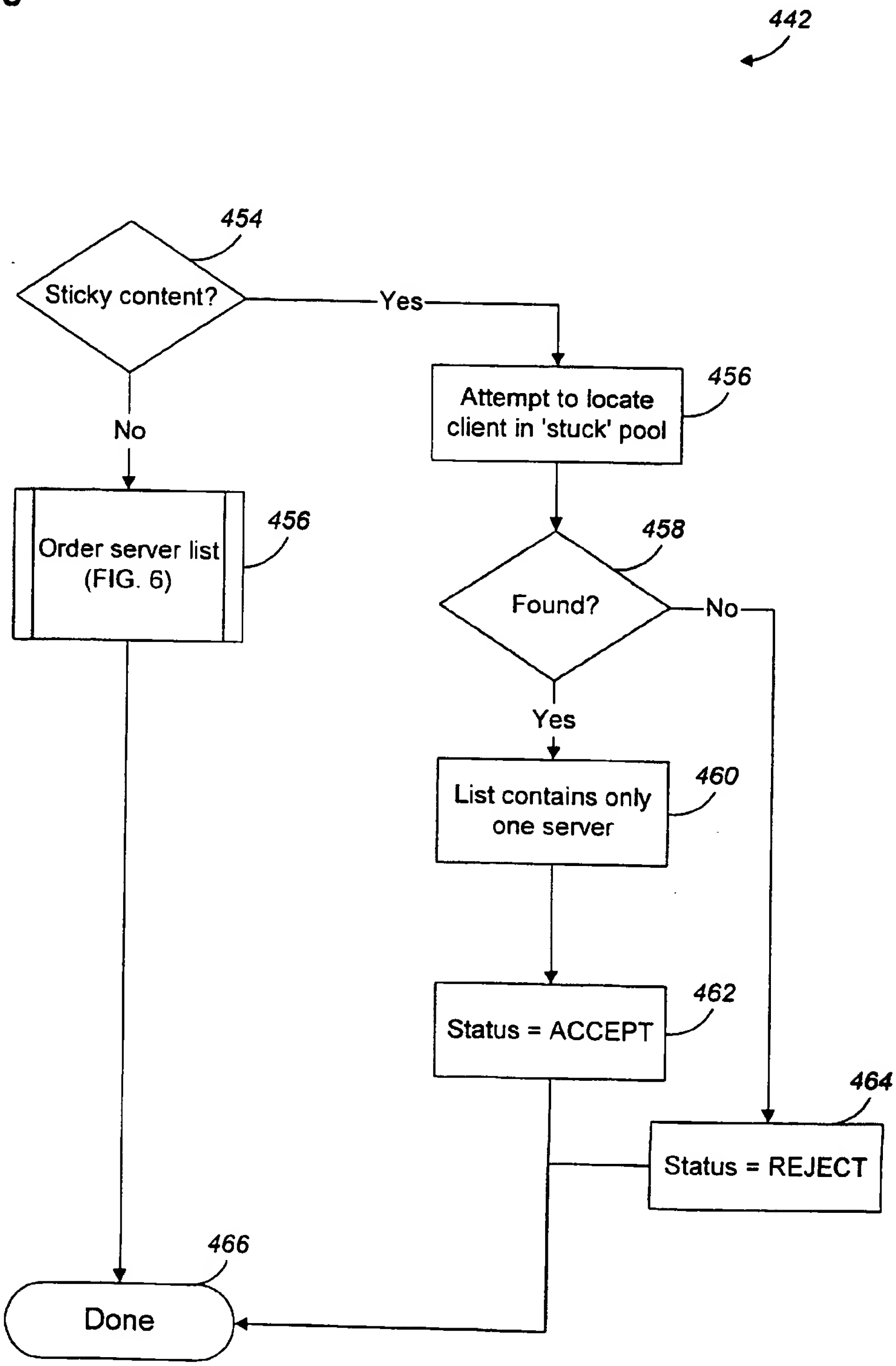

442
454
Sticky content?
Yes
No
456
Attempt to locate client in 'stuck' pool
Order server list (FIG. 6)
456
458
Found?
No
Yes
List contains only one server
460
Status = ACCEPT
462
464
Status = REJECT
466
Done

FIG. 6

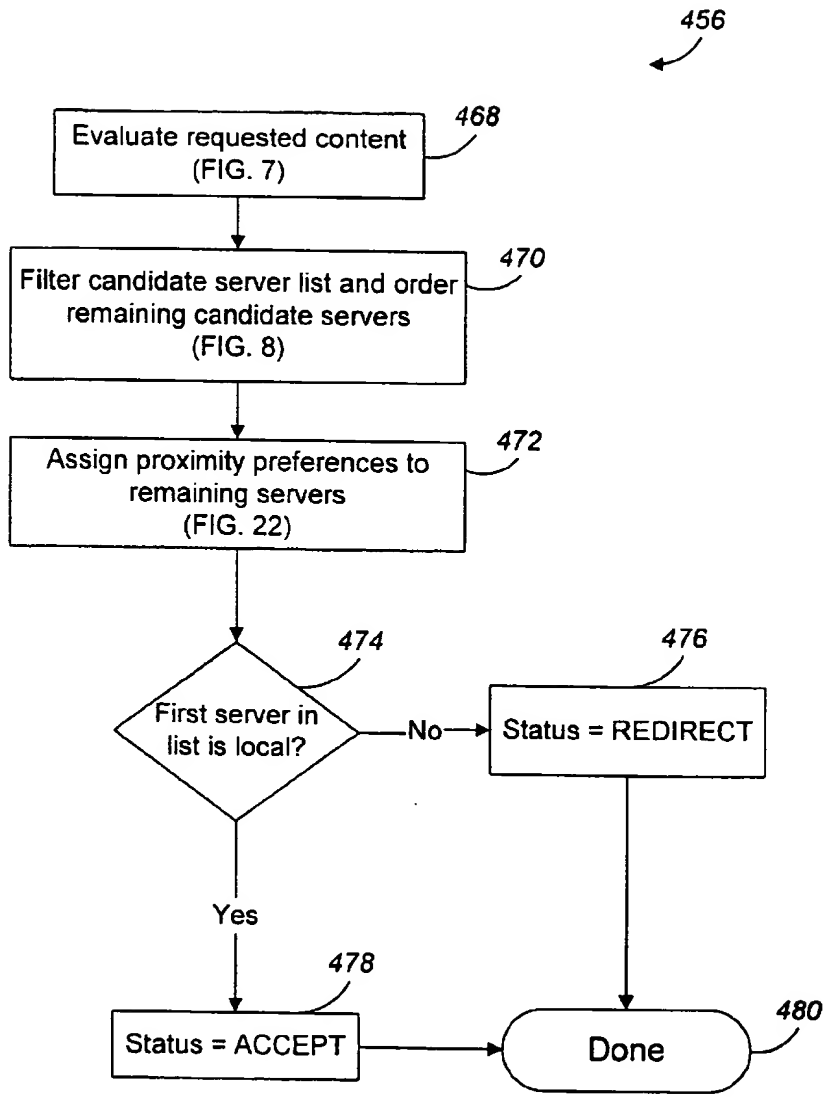

456
Evaluate requested content (FIG. 7)
468
Filter candidate server list and order remaining candidate servers (FIG. 8)
470
Assign proximity preferences to remaining servers (FIG. 22)
472
First server in list is local?
474
No
Status = REDIRECT
476
Yes
Status = ACCEPT
478
Done
480

FIG. 7

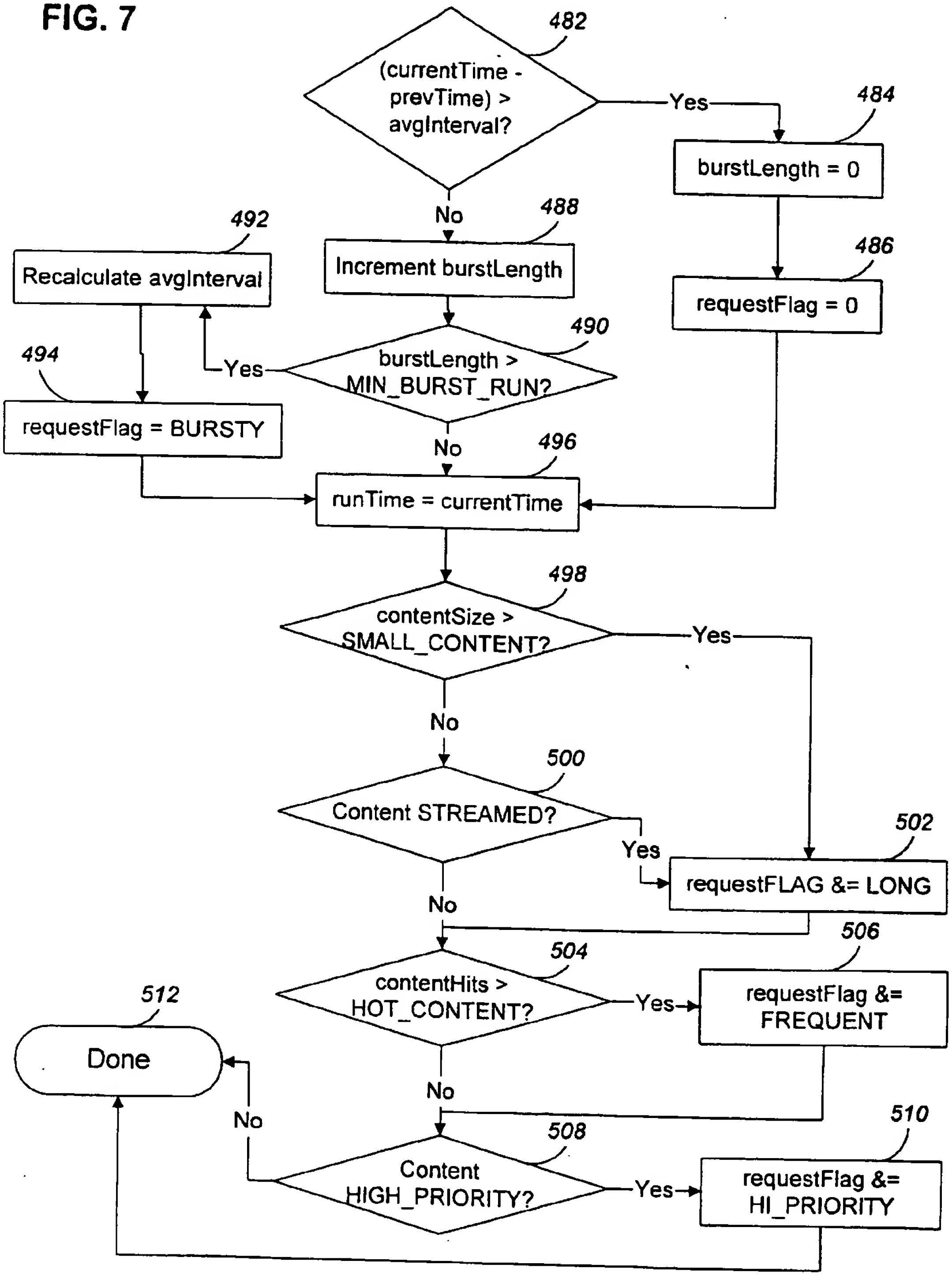

482
(currentTime - prevTime) > avgInterval?
Yes
No
484
burstLength = 0
486
requestFlag = 0
488
Increment burstLength
490
burstLength > MIN_BURST_RUN?
Yes
No
492
Recalculate avgInterval
494
requestFlag = BURSTY
496
runTime = currentTime
498
contentSize > SMALL_CONTENT?
Yes
No
500
Content STREAMED?
Yes
No
502
requestFLAG &= LONG
504
contentHits > HOT_CONTENT?
Yes
No
506
requestFlag &= FREQUENT
512
Done
508
Content HIGH_PRIORITY?
No
Yes
510
requestFlag &= HI_PRIORITY

FIG. 8

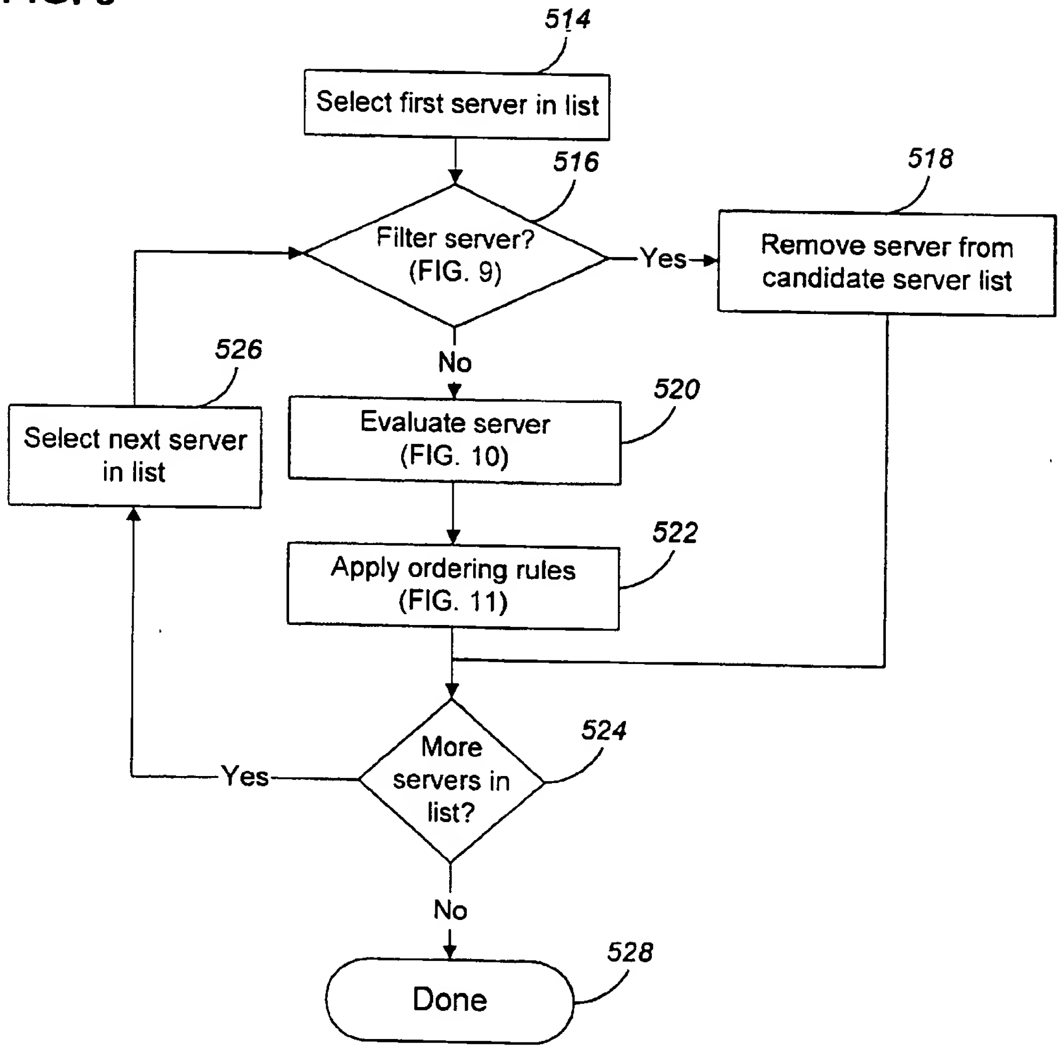

514
Select first server in list
516
Filter server? (FIG. 9)
Yes
518
Remove server from candidate server list
No
520
Evaluate server (FIG. 10)
522
Apply ordering rules (FIG. 11)
526
Select next server in list
524
More servers in list?
Yes
No
528
Done

FIG. 9

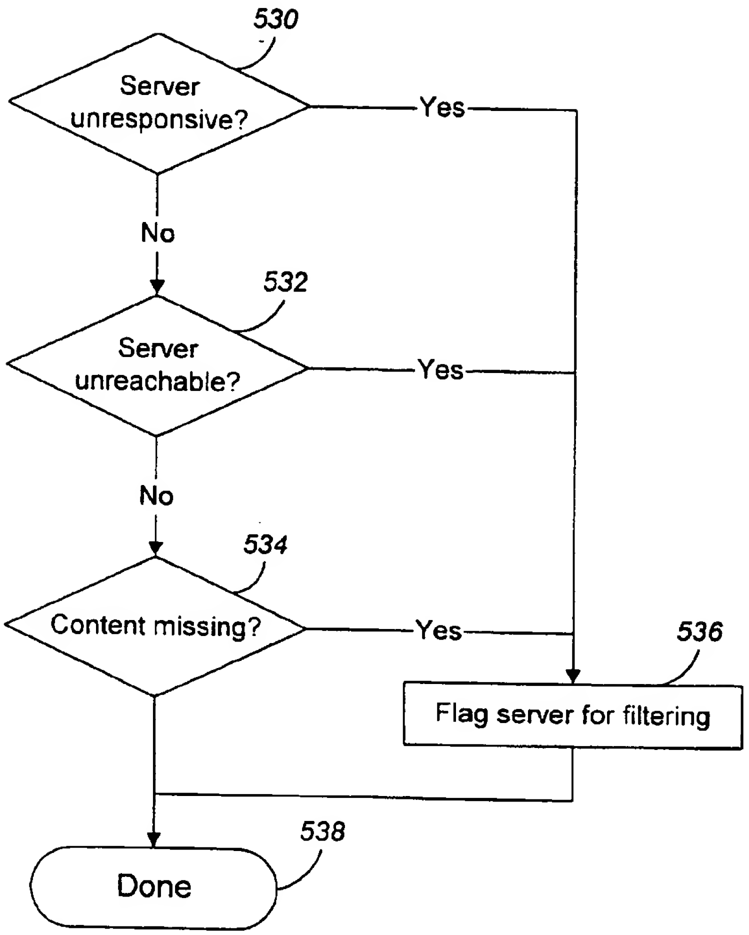

530
Server unresponsive?
Yes
No
532
Server unreachable?
Yes
No
534
Content missing?
Yes
536
Flag server for filtering
538
Done

FIG. 10

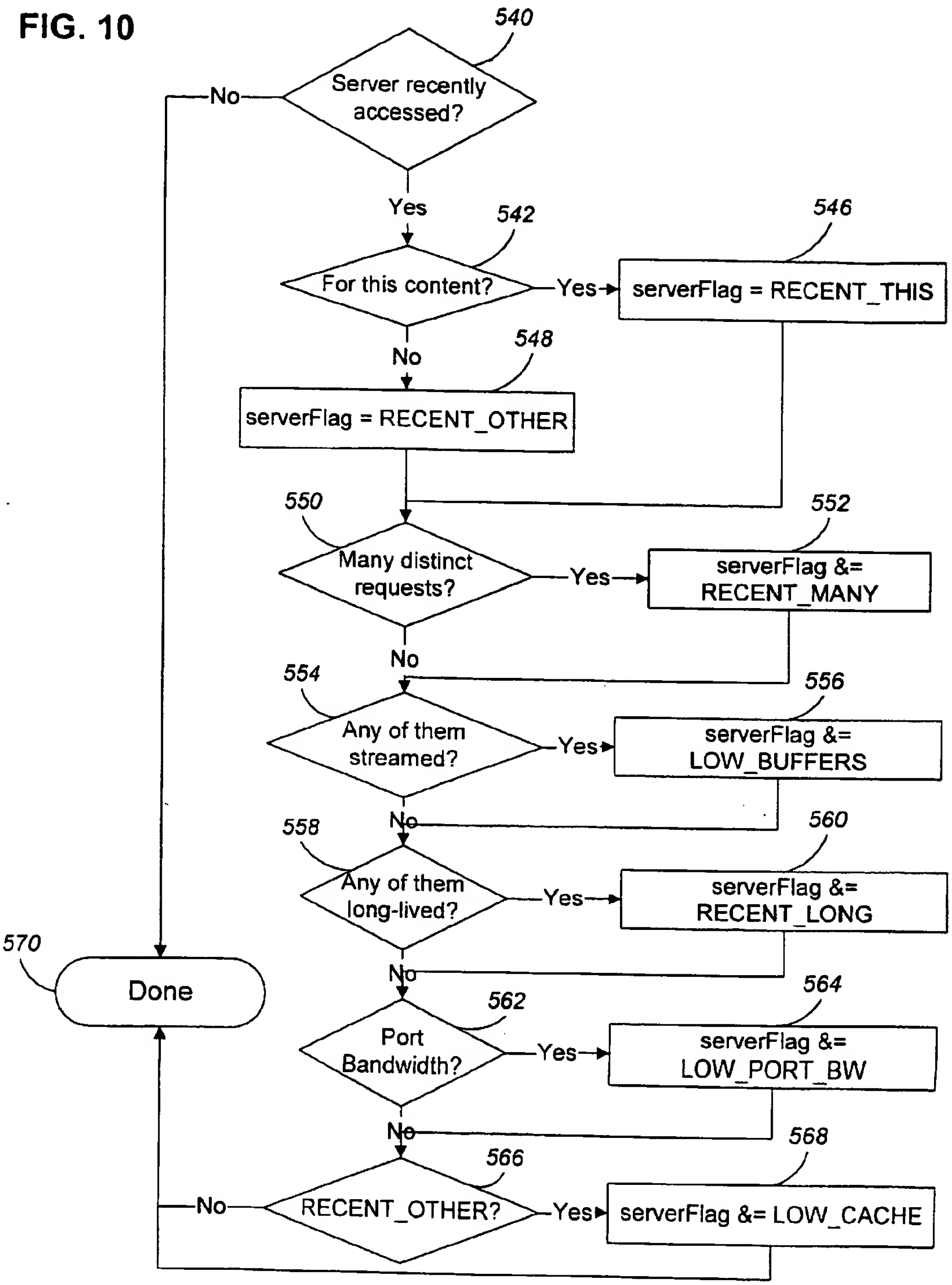

540
Server recently accessed?
No
Yes
542
For this content?
Yes
546
serverFlag = RECENT_THIS
No
548
serverFlag = RECENT_OTHER
550
Many distinct requests?
Yes
552
serverFlag &= RECENT_MANY
No
554
Any of them streamed?
Yes
556
serverFlag &= LOW_BUFFERS
No
558
Any of them long-lived?
Yes
560
serverFlag &= RECENT_LONG
570
Done
No
562
Port Bandwidth?
Yes
564
serverFlag &= LOW_PORT_BW
No
566
RECENT_OTHER?
No
Yes
568
serverFlag &= LOW_CACHE

FIG. 11

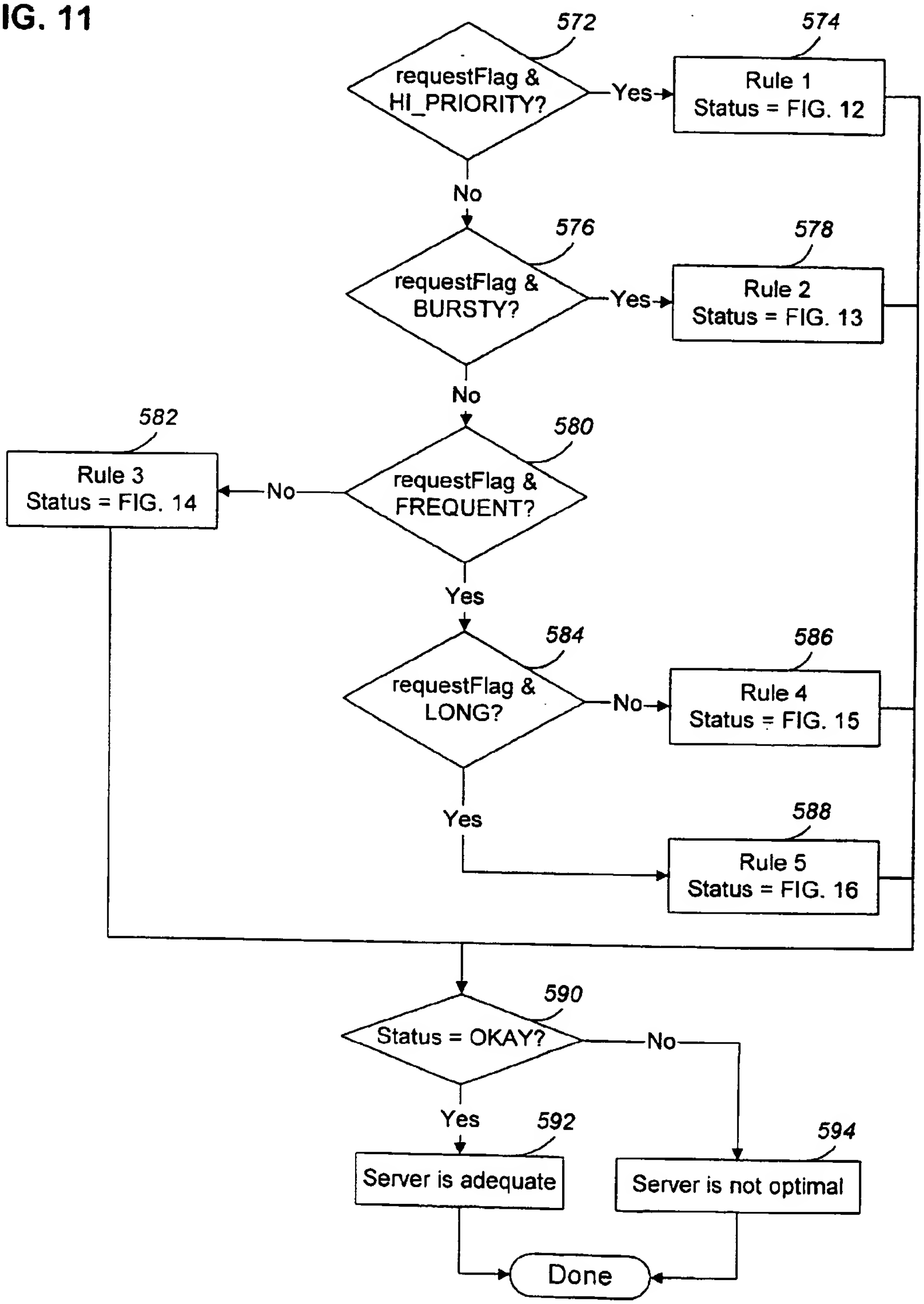

572
requestFlag & HI_PRIORITY?
Yes
574
Rule 1
Status = FIG. 12
No
576
requestFlag & BURSTY?
Yes
578
Rule 2
Status = FIG. 13
No
582
Rule 3
Status = FIG. 14
No
580
requestFlag & FREQUENT?
Yes
584
requestFlag & LONG?
No
586
Rule 4
Status = FIG. 15
Yes
588
Rule 5
Status = FIG. 16
590
Status = OKAY?
No
Yes
592
Server is adequate
594
Server is not optimal
Done

FIG. 12

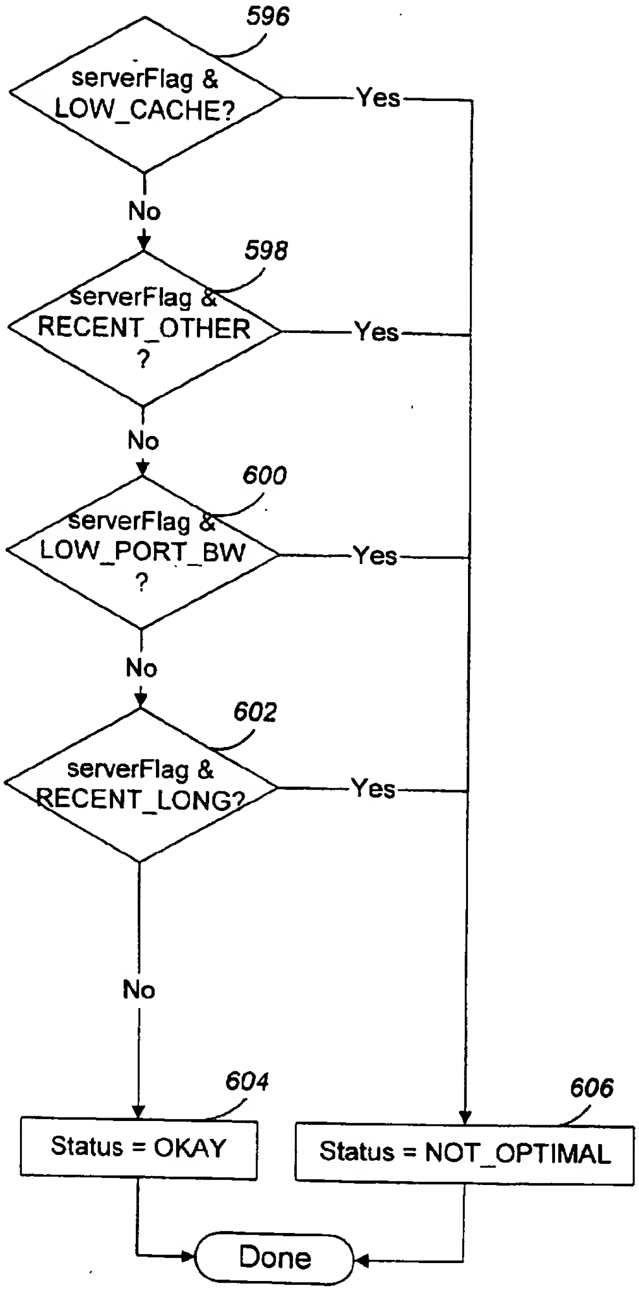

596
serverFlag & LOW_CACHE?
Yes
No
598
serverFlag & RECENT_OTHER ?
Yes
No
600
serverFlag & LOW_PORT_BW ?
Yes
No
602
serverFlag & RECENT_LONG?
Yes
No
604
Status = OKAY
606
Status = NOT_OPTIMAL
Done

FIG. 13

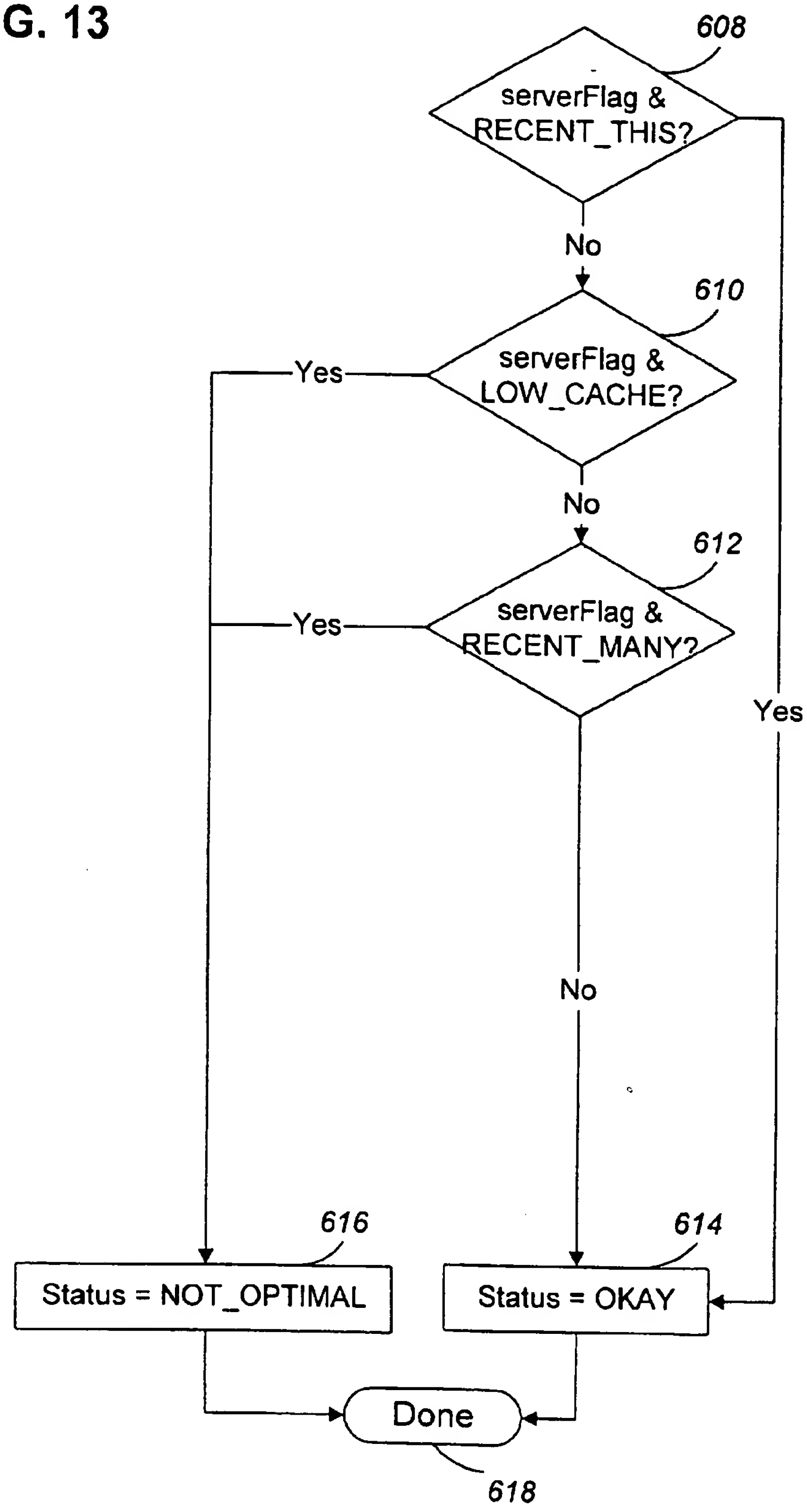

608
serverFlag & RECENT_THIS?
No
610
serverFlag & LOW_CACHE?
Yes
No
612
serverFlag & RECENT_MANY?
Yes
Yes
No
616
Status = NOT_OPTIMAL
614
Status = OKAY
Done
618

FIG. 14

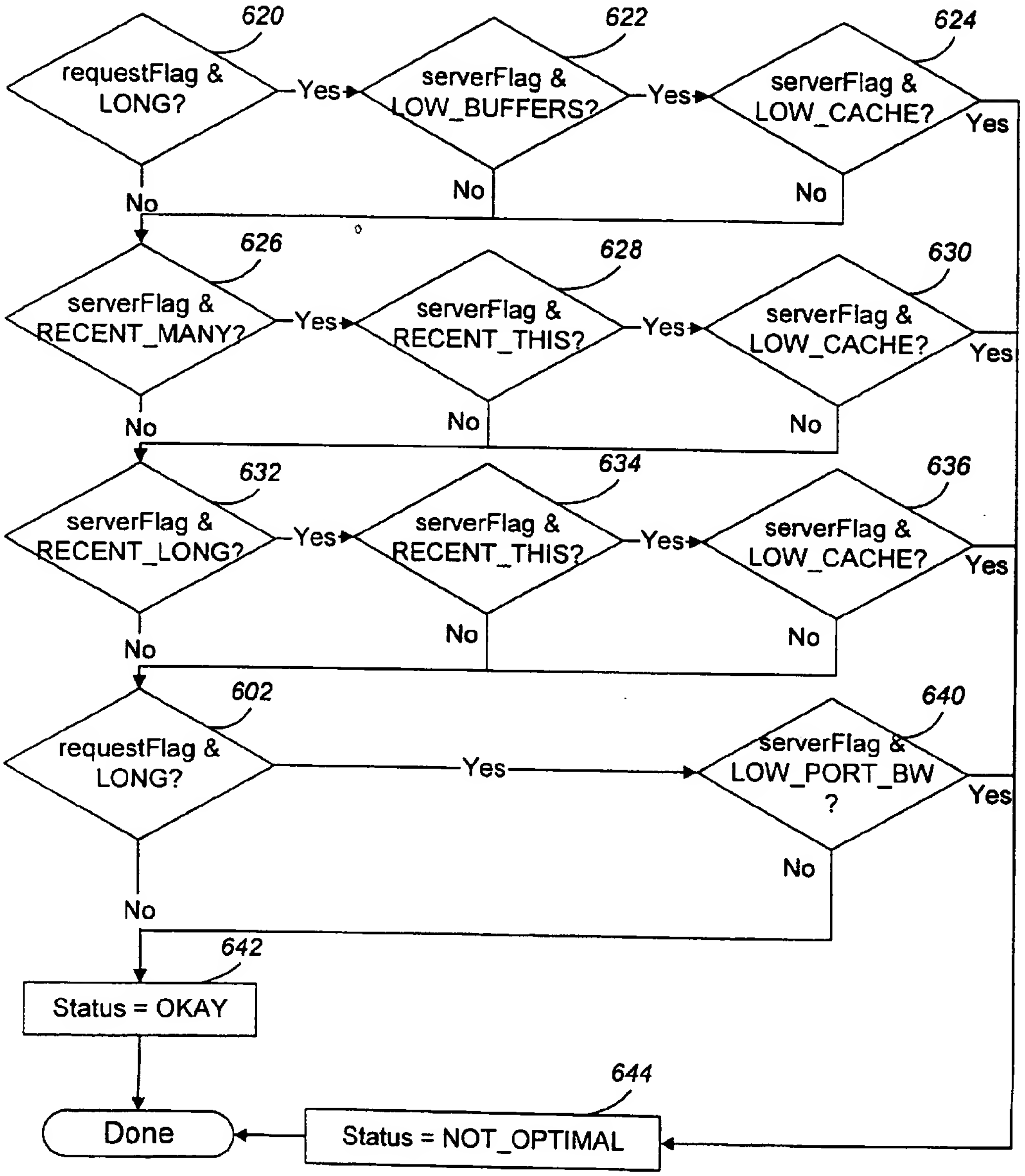

620
requestFlag & LONG?
Yes
622
serverFlag & LOW_BUFFERS?
Yes
624
serverFlag & LOW_CACHE?
Yes
No
No
No
626
serverFlag & RECENT_MANY?
Yes
628
serverFlag & RECENT_THIS?
Yes
630
serverFlag & LOW_CACHE?
Yes
No
No
No
632
serverFlag & RECENT_LONG?
Yes
634
serverFlag & RECENT_THIS?
Yes
636
serverFlag & LOW_CACHE?
Yes
No
No
No
602
requestFlag & LONG?
Yes
640
serverFlag & LOW_PORT_BW ?
Yes
No
No
642
Status = OKAY
644
Done
Status = NOT_OPTIMAL

FIG. 15

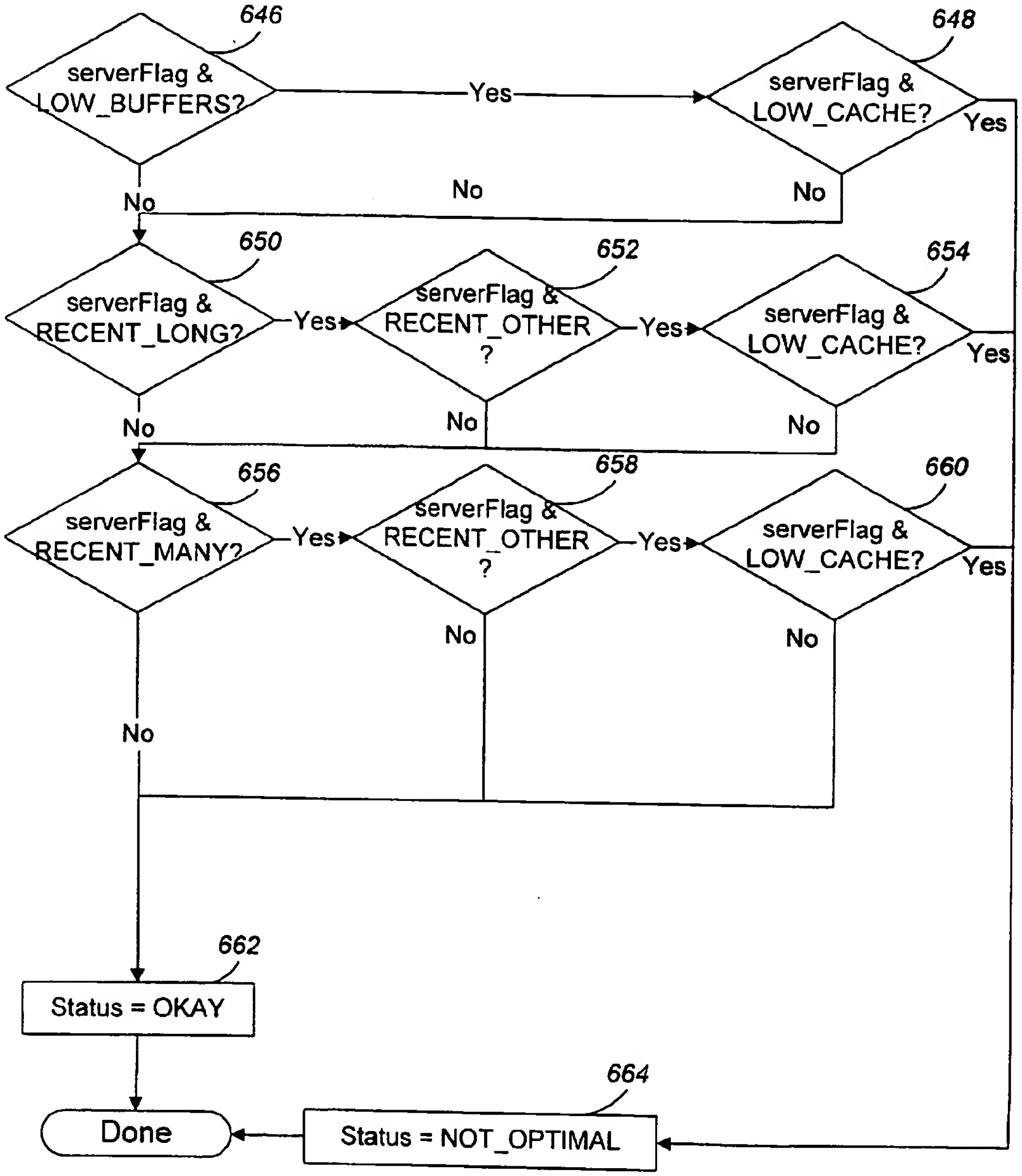

646
serverFlag & LOW_BUFFERS?
Yes
648
serverFlag & LOW_CACHE?
Yes
No
No
No
650
serverFlag & RECENT_LONG?
Yes
652
serverFlag & RECENT_OTHER ?
Yes
654
serverFlag & LOW_CACHE?
Yes
No
No
No
656
serverFlag & RECENT_MANY?
Yes
658
serverFlag & RECENT_OTHER ?
Yes
660
serverFlag & LOW_CACHE?
Yes
No
No
No
662
Status = OKAY
664
Done
Status = NOT_OPTIMAL

FIG. 16

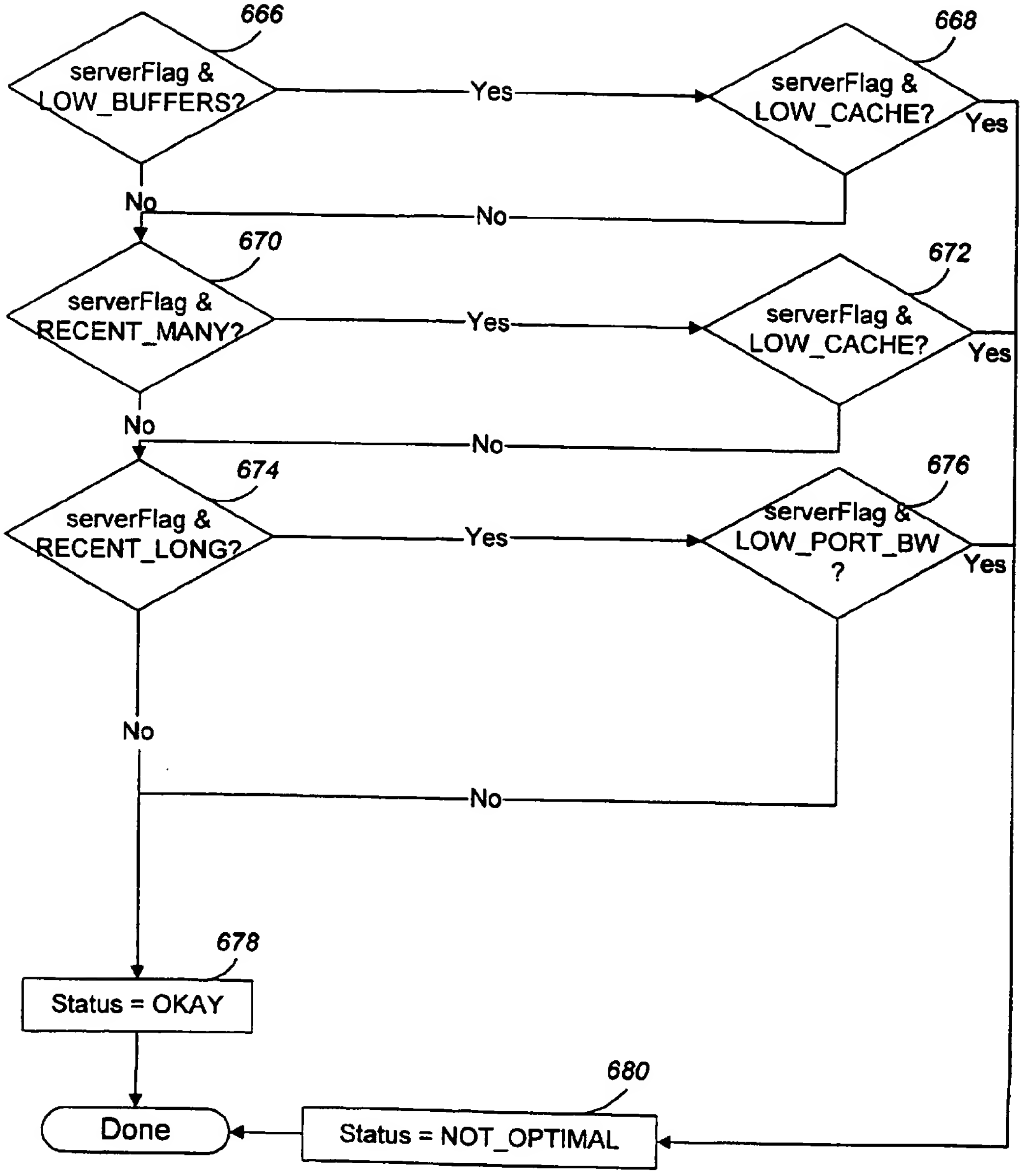

666
serverFlag & LOW_BUFFERS?
Yes
668
serverFlag & LOW_CACHE?
Yes
No
No
670
serverFlag & RECENT_MANY?
Yes
672
serverFlag & LOW_CACHE?
Yes
No
No
674
serverFlag & RECENT_LONG?
Yes
676
serverFlag & LOW_PORT_BW ?
Yes
No
No
678
Status = OKAY
Done
680
Status = NOT_OPTIMAL

FIG. 17

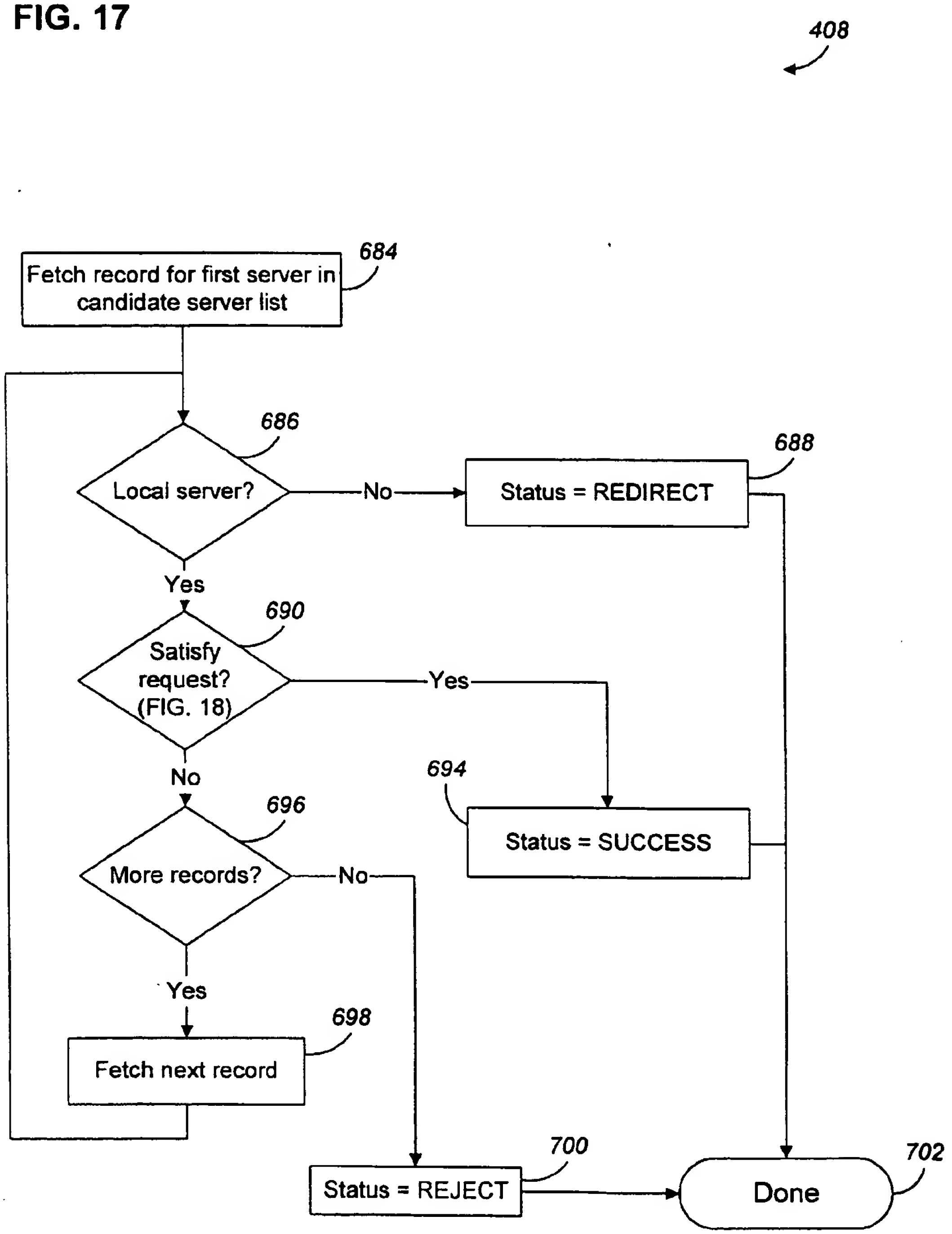

408
Fetch record for first server in candidate server list
684
686
Local server?
No
Status = REDIRECT
688
Yes
690
Satisfy request? (FIG. 18)
Yes
694
Status = SUCCESS
No
696
More records?
No
Yes
698
Fetch next record
700
Status = REJECT
702
Done

FIG. 18

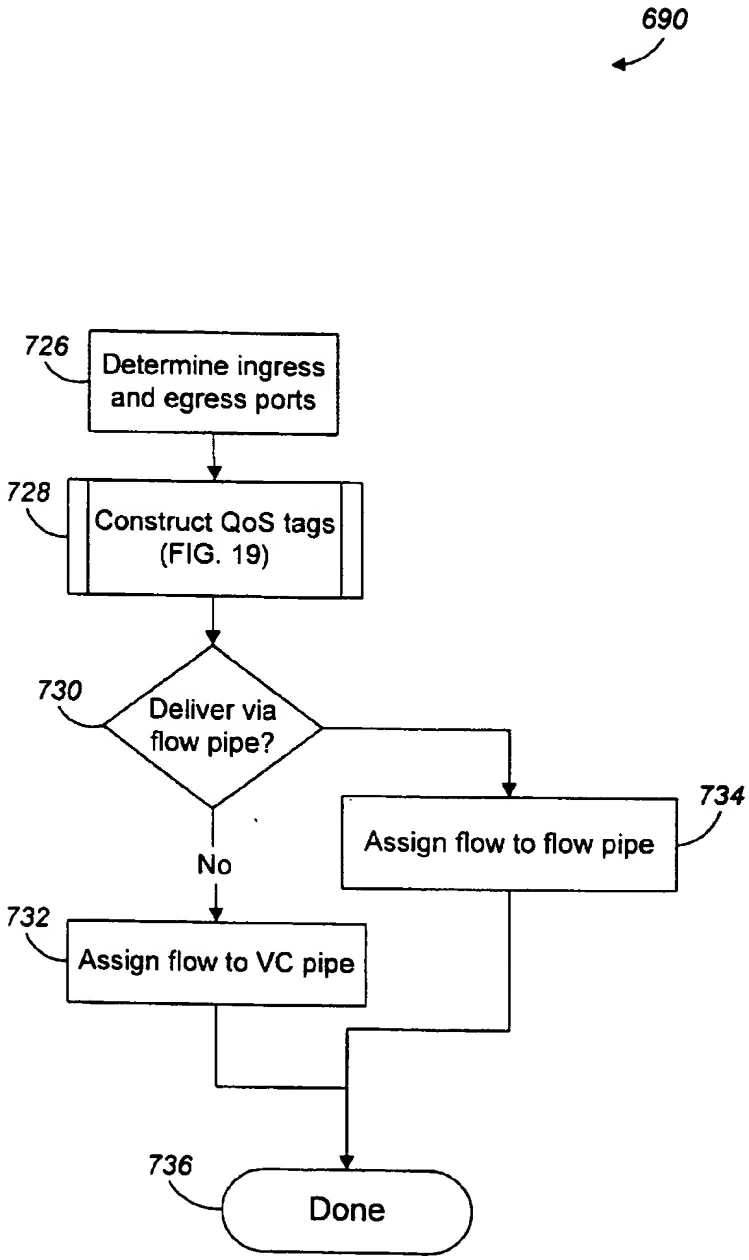

690
726
Determine ingress and egress ports
728
Construct QoS tags (FIG. 19)
730
Deliver via flow pipe?
No
734
Assign flow to flow pipe
732
Assign flow to VC pipe
736
Done

FIG. 19

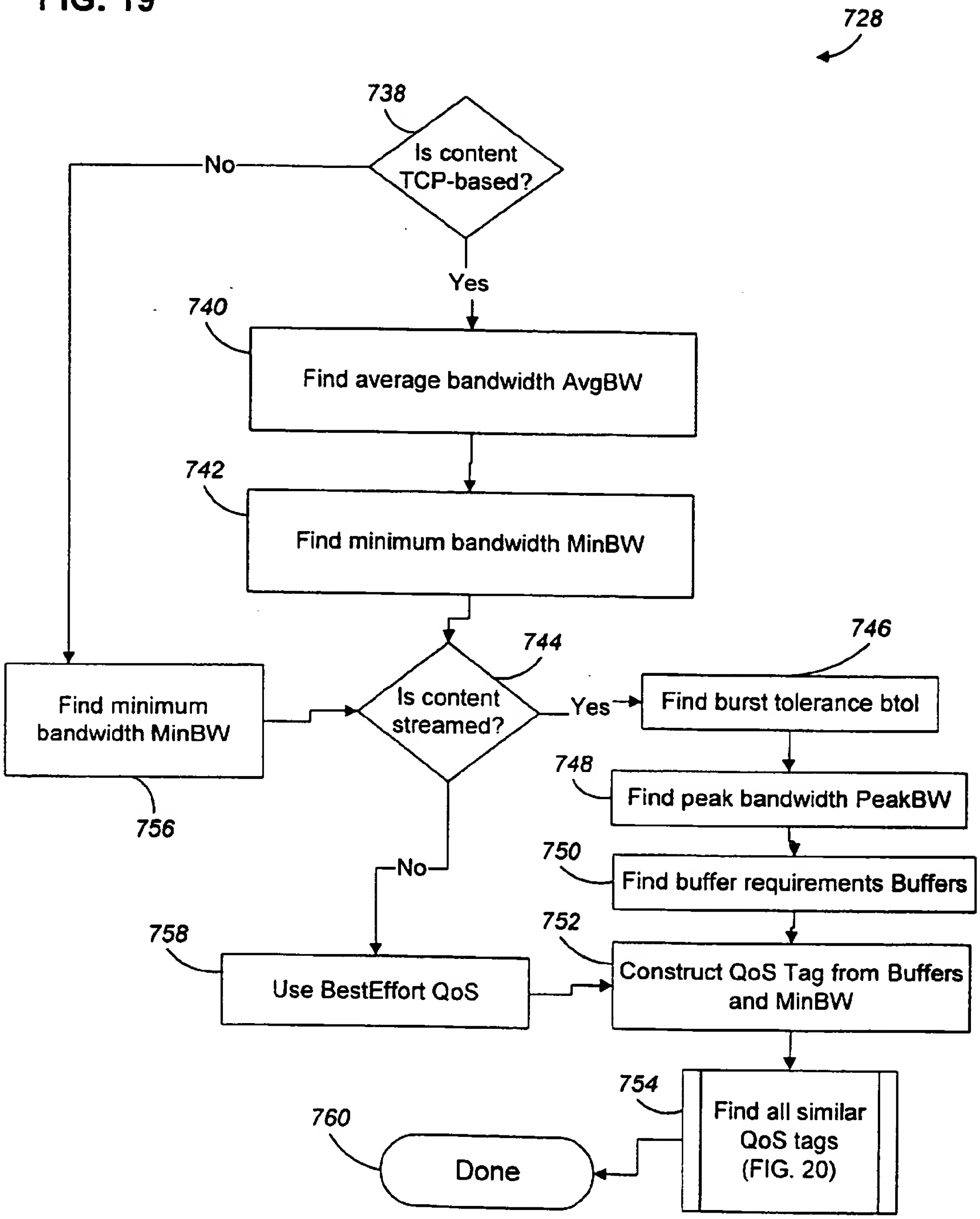

728
738
Is content TCP-based?
No
Yes
740
Find average bandwidth AvgBW
742
Find minimum bandwidth MinBW
744
Is content streamed?
746
Yes
Find burst tolerance btol
756
Find minimum bandwidth MinBW
748
Find peak bandwidth PeakBW
750
Find buffer requirements Buffers
No
758
Use BestEffort QoS
752
Construct QoS Tag from Buffers and MinBW
754
Find all similar QoS tags (FIG. 20)
760
Done

FIG. 20

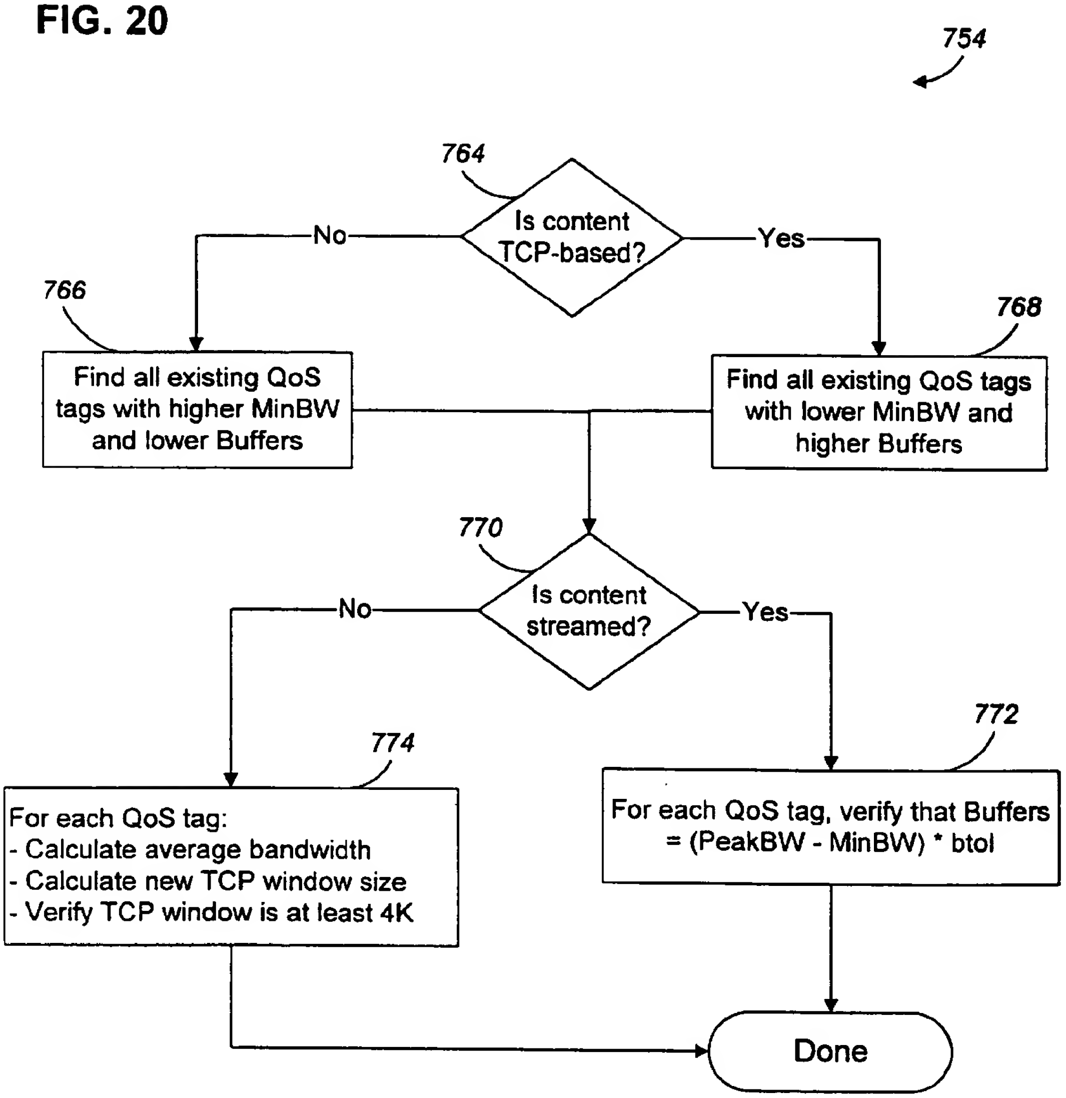
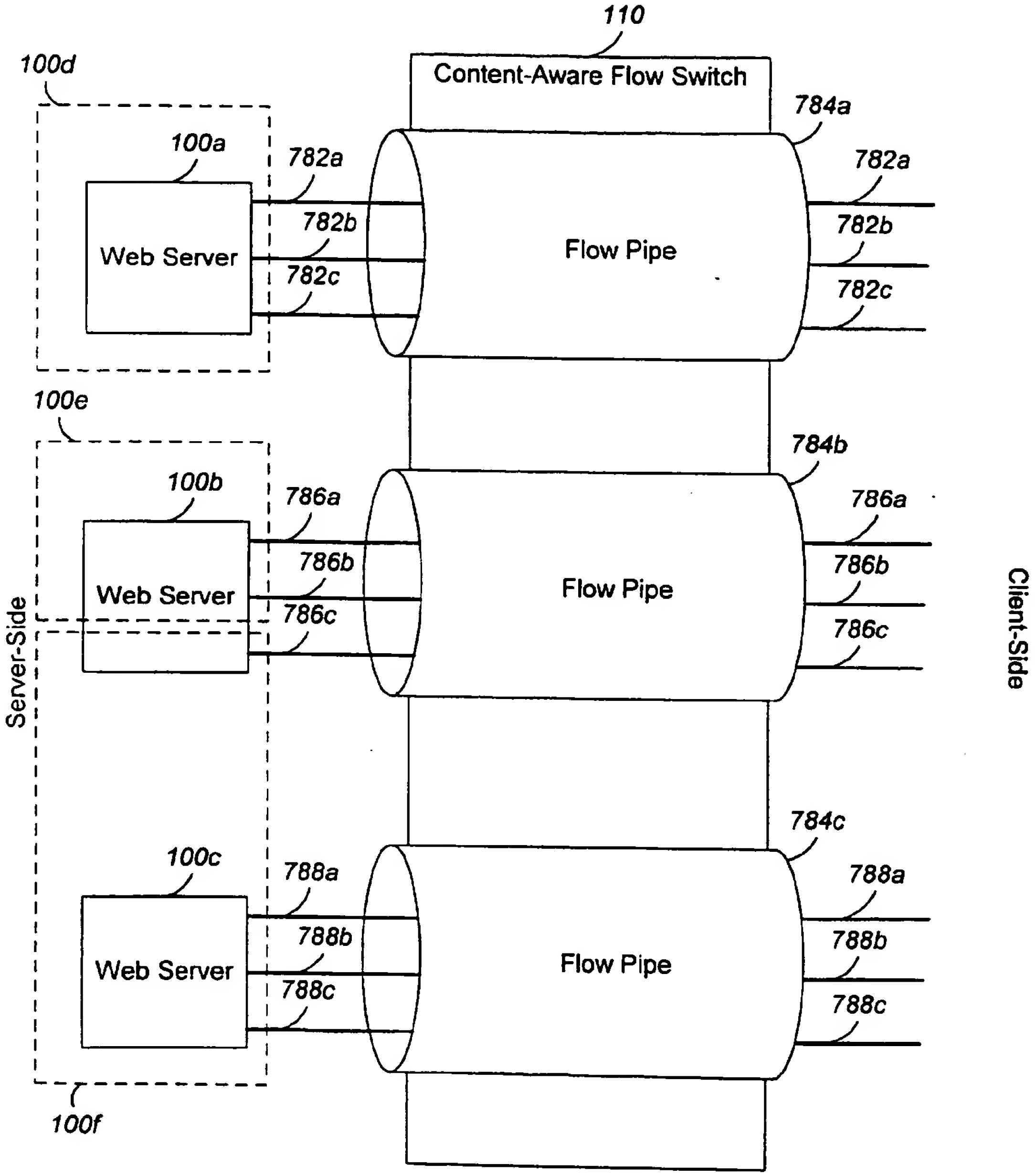

754
764
Is content TCP-based?
No
Yes
766
Find all existing QoS tags with higher MinBW and lower Buffers
768
Find all existing QoS tags with lower MinBW and higher Buffers
770
Is content streamed?
No
Yes
774
For each QoS tag:
- Calculate average bandwidth
- Calculate new TCP window size
- Verify TCP window is at least 4K
772
For each QoS tag, verify that Buffers = (PeakBW - MinBW) * btol
Done

FIG. 21a


110
Content-Aware Flow Switch
100d
100a
Web Server
782a
782b
782c
784a
782a
782b
782c
Flow Pipe
100e
100b
Web Server
786a
786b
786c
784b
786a
786b
786c
Flow Pipe
Server-Side
Client-Side
100c
Web Server
788a
788b
788c
784c
788a
788b
788c
Flow Pipe
100f

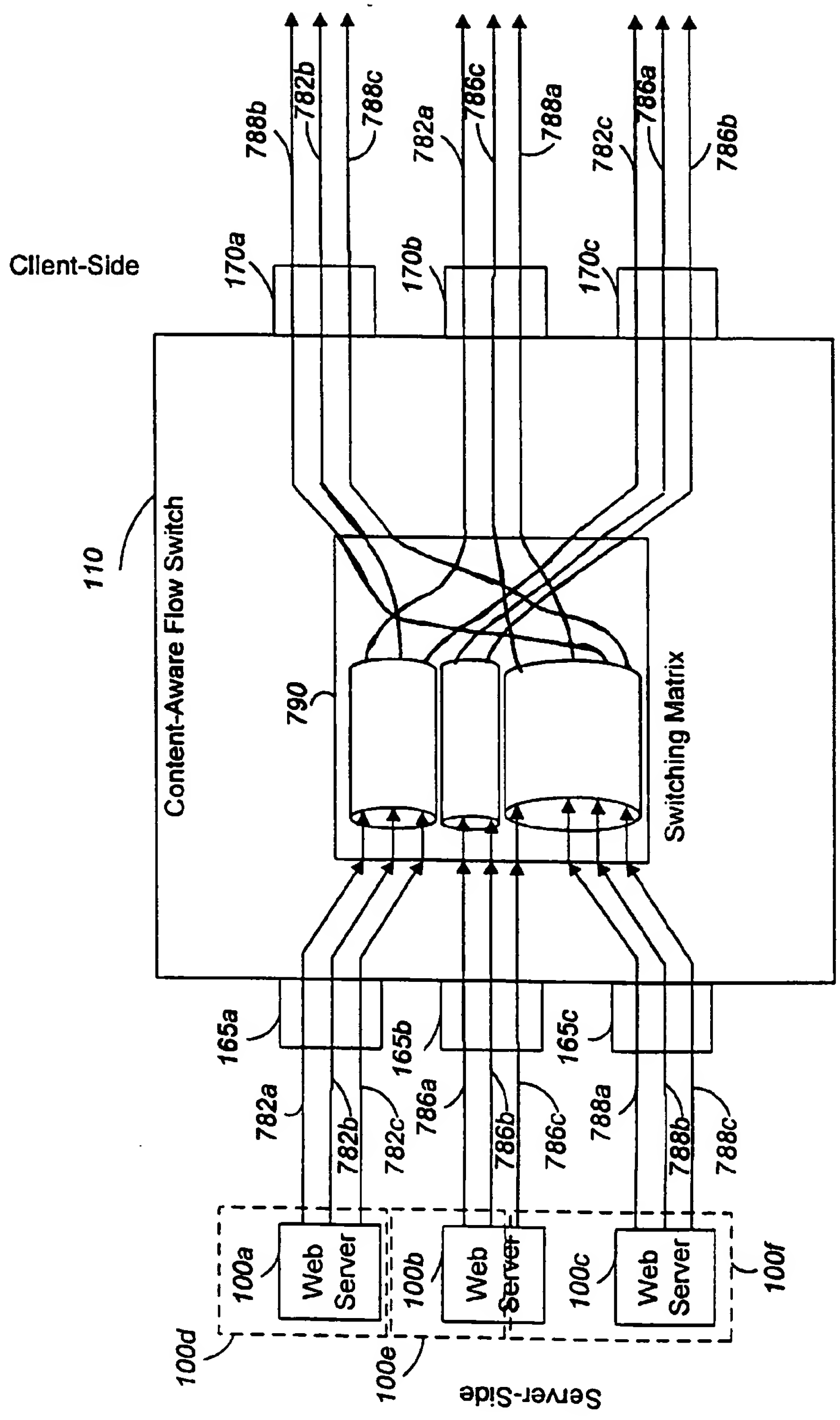

Client-Side
170a
170b
170c
788b
782b
788c
782a
786c
788a
782c
786a
786b
110
Content-Aware Flow Switch
790
Switching Matrix
165a
165b
165c
782a
782b
782c
786a
786b
786c
788a
788b
788c
100a
Web Server
100b
Web Server
100c
Web Server
100d
100e
100f
Server-Side

FIG. 21b

FIG. 22

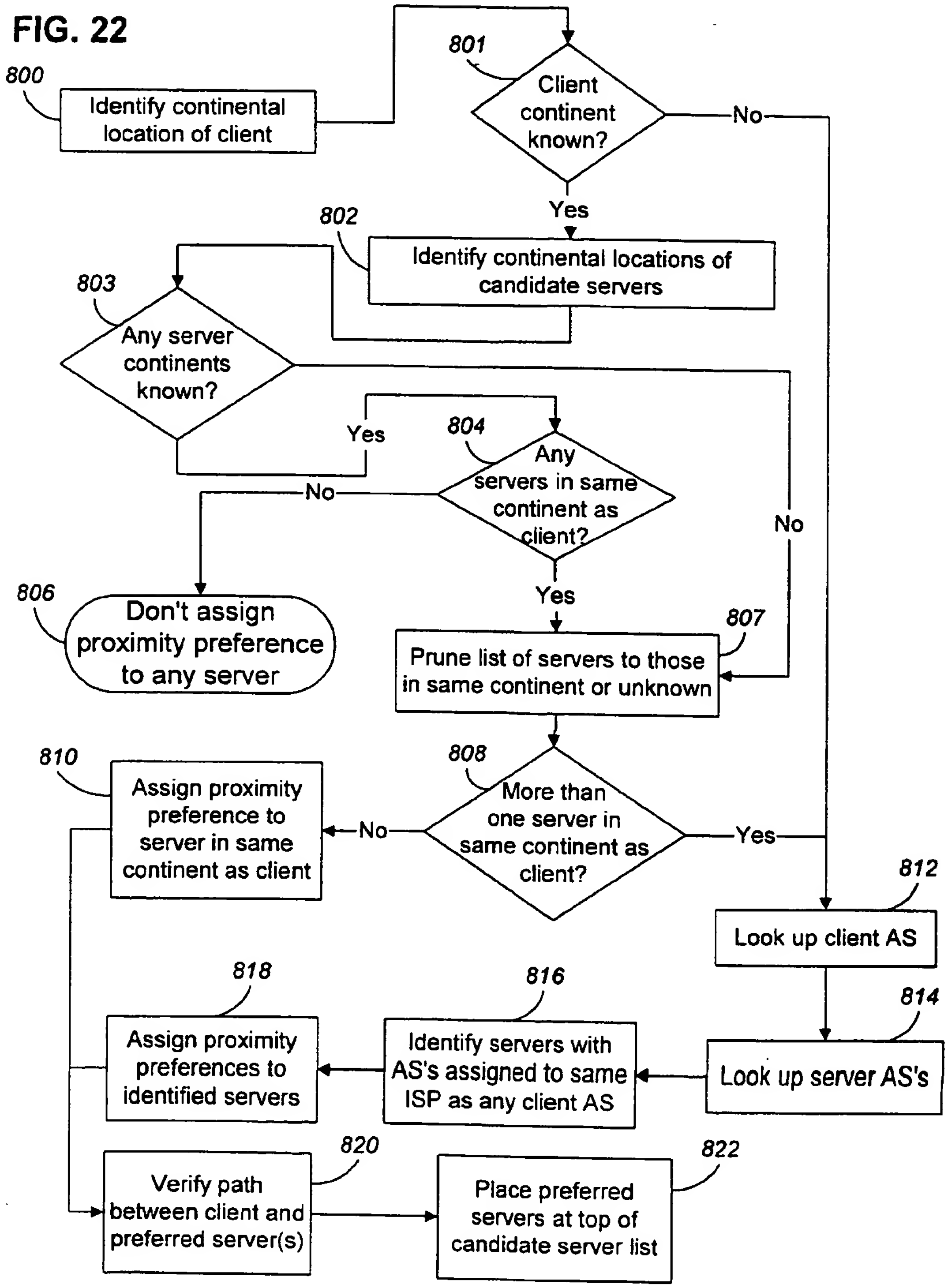
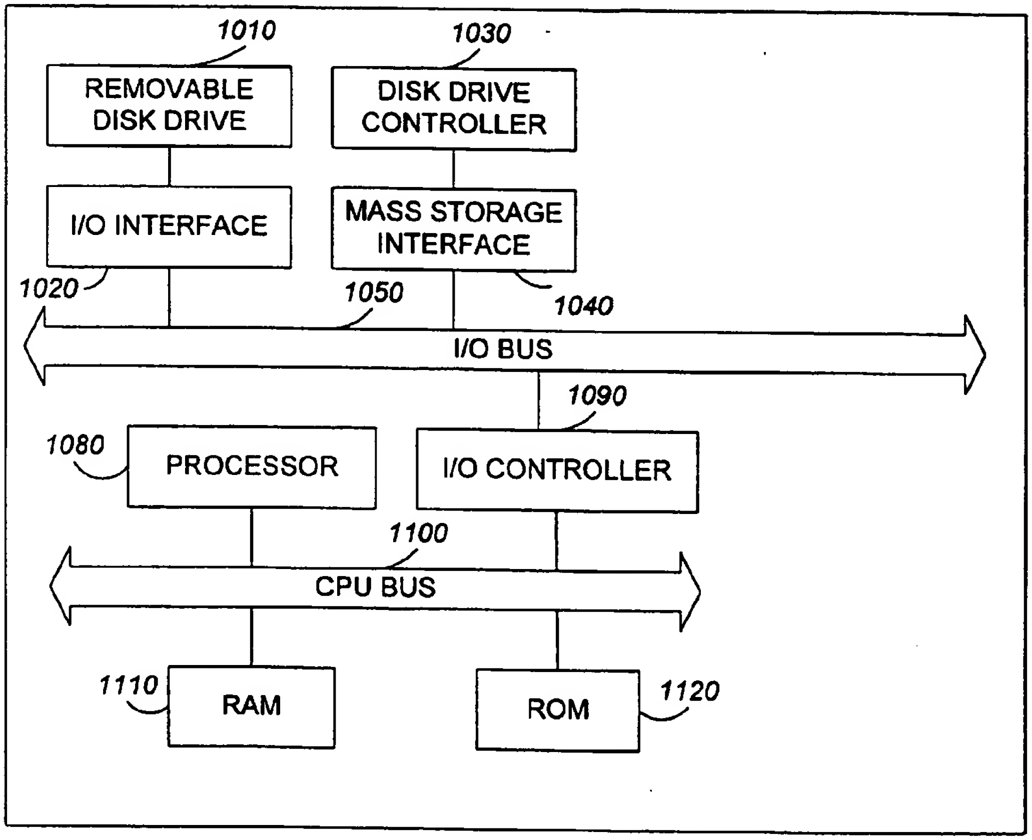

800
Identify continental location of client
801
Client continent known?
No
Yes
802
Identify continental locations of candidate servers
803
Any server continents known?
Yes
No
804
Any servers in same continent as client?
No
Yes
806
Don't assign proximity preference to any server
807
Prune list of servers to those in same continent or unknown
808
More than one server in same continent as client?
No
Yes
810
Assign proximity preference to server in same continent as client
812
Look up client AS
814
Look up server AS's
816
Identify servers with AS's assigned to same ISP as any client AS
818
Assign proximity preferences to identified servers
820
Verify path between client and preferred server(s)
822
Place preferred servers at top of candidate server list

FIG. 23


1010
REMOVABLE DISK DRIVE
1030
DISK DRIVE CONTROLLER
I/O INTERFACE
MASS STORAGE INTERFACE
1020
1050
1040
I/O BUS
1090
1080
PROCESSOR
I/O CONTROLLER
1100
CPU BUS
1110
RAM
ROM
1120

CONTENT-AWARE SWITCHING OF NETWORK PACKETS

This application is a Continuation of U.S. application Ser. No. 09/050,524, Filed Mar. 30, 1998, now issued U.S. Pat. No. 6,006,264, which claims priority from U.S. Provisional Application Ser. No. 60/054,687, Filed Aug. 1, 1997.

REFERENCES TO RELATED APPLICATIONS

This application claims priority from a provisional application Ser. No. 60/054,687, filed Aug. 1, 1997, which is hereby incorporated by reference.

BACKGROUND OF THE INVENTION

The present invention relates to content-based flow switching in Internet Protocol (IP) networks.

IP networks route packets based on network address information that is embedded in the headers of packets. In the most general sense, the architecture of a typical data switch consists of four primary components: (1) a number of physical network ports (both ingress ports and egress ports), (2) a data plane, (3) a control plane, and (4) a management plane. The data plane, sometimes referred to as the "fastpath," is responsible for moving packets from ingress ports of the data switch to egress ports of the data switch based on addressing information contained in the packet headers and information from the data switch's forwarding table. The forwarding table contains a mapping between all the network addresses the data switch has previously seen and the physical port on which packets destined for that address should be sent. Packets that have not previously been mapped to a physical port are directed to the control plane. The control plane determines the physical port to which the packet should be forwarded. The control plane is also responsible for updating the forwarding table so that future packets to the same destination may be forwarded directly by the data plane. The data plane functionality is commonly performed in hardware. The management plane performs administrative functions such as providing a user interface (UI) and managing Simple Network Management Protocol (SNMP) engines.

Packets conforming to the TCP/IP Internet layering model have 5 layers of headers containing network address information, arranged in increasing order of abstraction. A data switch is categorized as a layer N switch if it makes switching decisions based on address information in the Nth layer of a packet header. For example, both Local Area Network (LAN, layer 2) switching and IP (layer 3) switching switch packets based solely on address information contained in transmitted packet headers. In the case of LAN switching, the destination MAC address is used for switching, and in the case of IP switching, the destination IP address is used for switching.

Applications that communicate over the Internet typically communicate with each other over a transport layer (layer 4) Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) connection. Such applications need not be aware of the switching that occurs at lower levels (levels 1–3) to support the layer 4 connection. For example, an HyperText Transfer Protocol (HTTP) client (also known as a web browser) exchanges HTTP (layer 5) control messages and data (payload) with a target web server over a TCP (layer 4) connection.

"Content" can be loosely defined as any information that a client application is interested in receiving. In an IP network, this information is typically delivered by an application-layer server application using TCP or UDP as its transport layer. The content itself may be, for example, a simple ASCII text file, a binary file, an HTML page, a Java applet, or real-time audio or video.

A "flow" is a series of frames exchanged between two connection endpoints defined by a layer 3 network address and a layer 4 port number pair for each end of the connection. Typically, a flow is initiated by a request at one of the two connection endpoints for content which is accessible through the other connection endpoint. The flow that is created in response to the request consists of (1) packets containing the requested content, and (2) control messages exchanged between the two endpoints.

Flow classification techniques are used to associate priority codes with flows based on their Quality of Service (QoS) requirements. Such techniques prioritize network requests by treating flows with different QoS classes differently when the flows compete for limited network resources. Flows in the same QoS class are assigned the same priority code. A flow classification technique may, for example, classify flows based on IP addresses and other inner protocol header fields. For example, a QoS class with a particular priority may consist of all flows that are destined for destination IP address 142.192.7.7 and TCP port number 80 and TOS of 1 (Type of Service field in the IP header). This technique can be used to improve QoS by giving higher priority flows better treatment.

Internet Service Providers (ISPs) and other Internet Content Providers commonly maintain web sites for their customers. This service is called web hosting. Each web site is associated with a web host. A web host may be a physical web server. A web host may also be a logical entity, referred to as a virtual web host (VWH). A virtual web host associated with a large web site may span multiple physical web servers. Conversely, several virtual web hosts associated with small web sites may share a single physical web server. In either case, each virtual web host provides the functionality of a single physical web server in a way that is transparent to the client. The web sites hosted on a virtual web host share server resources, such as CPU cycles and memory, but are provided with all of the services of a dedicated web server. A virtual web host has one or more public virtual IP address that clients use to access content on the virtual web host. A web host is uniquely identified by its public IP address. When a content request is made to the virtual web host's virtual IP address, the virtual IP address is mapped to a private IP address, which points either to a physical server or to a software application identified by both a private IP address and a layer 4 port number that is allocated to the application.

SUMMARY OF THE INVENTION

In one aspect, the invention features content-aware flow switching in an IP network. Specifically, when a client in an IP network makes a content request, the request is intercepted by a content-aware flow switch, which seamlessly forwards the content request to a server that is well-suited to serve the content request. The server is chosen by the flow switch based on the type of content requested, the QoS requirements implied by the content request, the degree of load on available servers, network congestion information, and the proximity of the client to available servers. The entire process of server selection is transparent to the client.

In another aspect, the invention features implicit deduction of the QoS requirements of a flow based on the content of the flow request. After a flow is detected, a QoS category is associated with the flow, and buffer and bandwidth resources consistent with the QoS category of the flow are allocated. Implicit deduction of the QoS requirements of incoming flow requests allows network applications to significantly improve their Quality of Service (QoS) behavior by (1) preventing over-allocation of system resources, and (2) enforcing fair competition among flows for limited system resources based on their QoS classes by using a strict priority and weighted fair queuing algorithm.

In another aspect, the invention features flow pipes, which are logical pipes through which all flows between virtual web hosts and clients travel. A single content-aware flow switch can support multiple flow pipes. A configurable percentage of the bandwidth of a content-aware flow switch is reserved for each flow pipe.

In another aspect, the invention features a method for selecting a best-fit server, from among a plurality of servers, to service a client request for content in an IP network. A location of the client is identified. A location of each of the plurality of servers is identified. Servers that are in the same location as the client are identified. A server from among the plurality of servers is selected as the best-fit server, using a method which assigns a proximity preference to the identified servers. The location of the client may be a continent in which the client resides. The location of each of the plurality of servers may be a continent in which the server resides. Servers that are in the same location as the client may be identified by identifying administrative authorities associated with the client based on its IP address, identifying, for each of the plurality of servers, administrative authorities associated with the server, and identifying servers associated with an administrative authority that is associated with the client. The administrative authorities may be Internet Service Providers.

One advantage of the invention is that content-aware flow switches can be interconnected and overlaid on top of an IP network to provide content-aware flow switching regardless of the underlying technology used by the IP network. In this way, the invention provides content-aware flow switching without requiring modifications to the core of existing IP networks.

Another advantage of the invention is that by using content-aware flow switching, a server farm may gracefully absorb a content request spike beyond the capacity of the farm by directing content requests to other servers. This allows mirroring of critical content in distributed data centers, with overflow content delivery capacity and backup in the case of a partial communications failure. Content-aware flow switches also allow individual web servers to be transparently removed for service.

Another advantage of the invention is that it performs admission control on a per flow basis, based on the level of local network congestion, the system resources available on the content-aware flow switch, and the resources available on the web servers front-ended by the flow switch. This allows resources to be allocated in accordance with individual flow QoS requirements.

One advantage of flow pipes is that the virtual web host associated with a flow pipe is guaranteed a certain percentage of the total bandwidth available to the flow switch, regardless of the other activity in the flow switch. Another advantage of flow pipes is that the quality of service provided to the flows in a flow pipe is tailored to the QoS requirements implied by the content of the individual flows.

Another advantage of the invention is that, when performing server selection, a server in the same continent as the client is preferred over servers in another continent. Transcontinental network links introduce delay and are frequently congested. The server selection process tends to avoid such trans-continental links and the bottlenecks they introduce.

Another advantage of the invention is that, when performing server selection, a server that shares a "closest" backbone ISP with the client is preferred. Backbone ISPs connect with one another at Network Access Points (NAP). NAPs frequently experience congestion. By selecting a path between a client and a server that does not include a NAP, bottlenecks are avoided.

Other features and advantages of the invention will become apparent from the following description and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1*a* is a block diagram of an IP network.

FIG. 1*b* is a block diagram of a segment of a network employing a content-aware flow switch.

FIG. 1*c* is a block diagram of traffic flow through a content-aware flow switch.

FIG. 2 is a block diagram illustrating operations performed by and communications among components of a content-aware flow switch during flow setup.

FIG. 3 is a flow chart of a method for servicing a content request using a content-aware flow switch.

FIG. 4 is a flow chart of a method for parsing a flow setup request.

FIGS. 5 and 6 are flow charts of methods for sorting a list of candidate servers.

FIG. 7 is a flow chart of a method for evaluating requested content.

FIG. 8 is a flow chart of a method for sorting a list of candidate servers.

FIG. 9 is a flow chart of a method for filting servers from a list of candidate servers.

FIG. 10 is a flow chart of a method for evaluating a server in a list of candidate servers.

FIG. 11 is a flow chart of a method for ordering a server in a list of candidate servers.

FIGS. 12–16 are flows charts of methods for assigning a status to a server for purposes of ordering the server in a list of candidate servers.

FIG. 17 is a flow chart of a method for assigning a flow to a local server.

FIG. 18 is a flow chart of a method for attempting to satisfy a request for a flow.

FIG. 19 is a flow chart of a method for constructing a QoS tag.

FIG. 20 is a flow chart of a method for locating QoS tags which are similar to a given QoS tag.

FIGS. 21*a–b* are block diagrams of flow pipe traffic through a content-aware flow switch.

FIG. 22 is a flow chart of a method for ordering servers in a list of candidate servers based on proximity.

FIG. 23 is a block diagram of a computer and computer elements suitable for implementing elements of the invention.

DETAILED DESCRIPTION

Referring to FIG. 1*a*, in a conventional IP network 100, such as the Internet, servers are connected to routers at the edges of the network **100**. Each router is connected to one or more other routers. Each stream of information transmitted from one end station to another is broken into packets containing, among other things, a destination address indicating the end station to which the packet should be delivered. A packet is transmitted from one end station to another via a sequence of routers. For example, a packet may originate at server S1, traverse routers R1, R2, R3, and R4, and then be delivered to server S2.

In FIG. **1***a*, a network node is either a router or an end station. Each router has access to information about each of the nodes to which the router is connected. When a router receives a packet, the router examines the packet's destination address, and forwards the packet to a node that the router calculates to be most likely to bring the packet closer to its destination address. The process of choosing an intermediary destination for a packet and forwarding the packet to the intermediary destination is called routing.

For example, referring to FIG. **1***a*, server S1 transmits a packet, whose destination address is server S2, to router R1. Router R1 is only connected to server S1 and to router R2. Router R1 therefore forwards the packet to router R2. When the packet reaches router R2, router R2 must choose to forward the packet to one of routers R1, R5, R3, and R6 based on the packet's destination IP address. The packet is passed from router to router until it reaches its destination of server S2.

Referring to FIG. **1***b*, web servers **100***a–c* and **120***a–b* are connected to a content-aware flow switch **110**. The web servers **100***a–c* are connected to the flow switch **110** over LAN links **105***a–c*. The web servers **120***a–b* are connected to the flow switch **110** over WAN links **122***a–b*. The flow switch **110** may be configured and its health monitored using a network management station **125**. The role of the management station **125** is to control and manage one or more communications devices from an external device such as a workstation running network management applications. The network management station **125** communicates with network devices via a network management protocol such as the Simple Network Management Protocol (SNMP). The flow switch **110** may connect to the network **100** (FIG. **1***a*) through a router **130**. The flow switch **110** is connected to the router **130** by a LAN or WAN link **132**. Alternatively, the flow switch **110** may connect to the network **100** directly via one or more WAN links (not shown). The router **130** connects to an Internet Service Provider (ISP) (not shown) by multiple WAN links **135***a–c*.

Referring to FIG. **1***c*, a content-aware flow switch "frontends" (i.e., intercepts all packets received from and transmitted by) a set of local web servers **100***a–c*, constituting a web server farm **150**. Although connections to the web servers **100***a–c* are typically initiated by clients on the client side, most of the traffic between a client and the server farm **150** is from the servers **100***a–c* to the client (the response traffic). It is this response traffic that needs to be most carefully controlled by the flow switch **110**.

The flow switch **110** has a number of physical ingress ports **170***a–c* and physical egress ports **165***a–c*. Each of the physical ingress ports **170***a–c* may act as one or more logical ingress ports, and each of the physical egress ports **165***a–c* may act as one or more logical egress ports in the procedures described below. Each of the web servers **100***a–c* is network accessible to the content-aware flow switch **110** via one or more of the physical egress ports **165***a–c*. Associated with each flow controlled by the flow switch **110** is a logical ingress port and a logical egress port.

The flow switch **110** is connected to an internet through uplinks **155***a–c*. When a client content request is accepted by the flow switch **110**, the flow switch **110** establishes a full-duplex logical connection between the client and one of the web servers **100***a–c* through the flow switch **110**. Individual flows are aggregated into pipes, as described in more detail below. Request traffic flows from the client toward the server and response traffic flows from the server to the client. A component of the flow switch **110**, referred to as the Flow Admission Control (FAC), polices if and how flows are admitted to the flow switch **110**, as described in more detail below.

The content-aware flow switch **110** differs from typical layer **2** and layer **3** switches in several respects. First, the data plane of layer **2** and layer **3** switches forwards packets based on the destination addresses in the packet headers (the MAC address and header information in the case of a layer **2** switch and the destination IP address in the case of a layer **3** switch). The content-aware flow switch **110** switches packets based on a combination of source and destination IP addresses, transport layer protocol, and transport layer source and destination port numbers. Furthermore, the functions performed in the control plane of typical layer **2** and layer **3** switches are based on examination of the layer **2** and layer **3** headers, respectively, and on well-known bridging and routing protocols. The control plane of the content-aware flow switch **110** also performs these functions, but additionally derives the forwarding path from information contained in the packet headers up to and including layer **5**. In addition, content-induced QoS and bandwidth requirements, server loading and network path optimization are also considered by the content-aware flow switch **110** when selecting the most optimal path for a packet, as described in more detail below.

FIG. **2** is a block diagram illustrating, at a high level, operations performed by and communications among components of the content-aware flow switch **110** during flow setup. An arrow between two components in FIG. **2** indicates that communication occurs in the direction of the arrow between the two components connected by the arrow.

Referring to FIG. **2**, the content-aware flow switch **110** includes: a Web Flow Redirector (WFR), an Intelligent Content Probe (ICP), a Content Server Database (CSD), a Client Capability Database (CCD), a Flow Admission Control (FAC), an Internet Probe Protocol (IPP), and an Internet Proximity Assist (IPA).

The CSD maintains several databases containing information about content flow characteristics, content locality, and the location of and the load on servers, such as servers **100***a–c* and **120***a–b*. One database maintained by the CSD contains content rules, which are defined by the system administrator and which indicate how the flow switch **110** should handle requests for content. Another database maintained by the CSD contains content records which are derived from the content rules. Content records contain information related to particular content, such as its associated IP address, URL, protocol, layer **4** port number, QoS indicators, and the load balance algorithm to use when accessing the content. A content record for particular content also points to server records identifying servers containing the particular content. Another database maintained by the CSD contains server records, each of which contains information about a particular server. The server record for a server contains, for example, the server's IP address, protocol, a port of the server through which the server can be accessed by the flow switch **110**, an indication of whether the server is local or remote with respect to the flow switch **110**, and load metrics indicating the load on the server.

Information in the CSD is periodically updated from various sources, as described in more detail below. The WFR, CSD, and FAC are responsible for selecting a server to service a content request based on a variety of criteria. The FAC uses server-specific and content-specific information together with client information and QoS requirements to determine whether to admit a flow to the flow switch **110**. The ICP is a lightweight HTTP client whose job is to populate the CSD with server and content information by probing servers for specific content that is not found in the CSD during a flow setup. The ICP probes servers for several reasons, including: (1) to locate specific content that is not already stored in the CSD, (2) to determine the characteristics of known content such as its size, (3) to determine relationships between different pieces of content, and (4) to monitor the health of the servers. ICPs on various flow switches communicate with each other using the IPP, which periodically sends local server load and content information to neighboring content-aware flow switches. The CCD contains information related to the known capabilities of clients and is populated by sampling specific flows in progress. The IPA periodically updates the CSD on the internet proximity of servers and clients.

A flow setup request may take the form of a TCP SYN from a client being forwarded to the WFR (**202**). The WFR passes the flow setup request to the CSD (**204**). The CSD determines which servers, if any, are available to service the flow request and generates a list of such candidate servers (**206**). This list of candidate servers is ordered based on configurable CSD preferences. The individual items within this list contain all the information the FAC will ultimately need to make flow admission decisions.

If more than one server exists in the server farm **150** and content is not fully replicated among the servers in the server farm, then it may not be possible for the CSD to identify any candidate servers based upon the receipt of the TCP SYN alone. In this case, the CSD returns a NULL candidate server list to the WFR with a status indicator requesting that the TCP connection is to be spoofed and that the subsequent HTTP GET is to be forwarded to the CSD (**212**).

If the CSD contains no content records for servers that can satisfy the received TCP SYN or HTTP GET, a NULL list is returned to the WFR with a status indicator indicating that the flow request should be rejected (**212**). If the CSD finds a content record that satisfies the HTTP GET but does not find a record for the specific piece of content requested, a new content record is created containing default values for the specific piece of content requested. The new record is then returned to the WFR (**212**). In either of these two cases (i.e., the CSD finds no matching records, or the CSD finds a matching record that does not exactly match the requested content), the CSD asks the ICP to probe the local servers (using http "HEAD" operations) to determine where the content is located and to deduce the content's QoS attributes (**208**).

The CSD then asks the CCD for information related to the client making the request (**211**). The CCD returns any such information in the CCD to the CSD (**210**). The CSD returns an ordered list of candidate servers and any client information obtained from the CCD to the WFR (**212**).

Depending on the response returned from the CSD, the WFR will either: (1) reject, TCP spoof, or redirect the flow as appropriate (**214**), or (2) forward the flow request, the list of candidate servers, and any client information to the FAC for selection and local setup (**216**). The FAC evaluates the list of servers contained in the content record, in the order specified by the CSD, and looks for a server that can accept the flow (**218**). The FAC's primary consideration in selecting a server from the list of candidate servers is that sufficient port and switch resources be available on the content-aware flow switch to support the flow. An accepted flow is assigned either to a VC-pipe or to a flow pipe, as appropriate. (VC-pipes and flow pipes are described in more detail below.) The FAC also adjusts flow weights as necessary to maintain flow pipe bandwidth.

The FAC informs the WFR of which local server, if any, was chosen to accept the flow, and provides information to the WFR indicating to which specific VC-pipe or flow pipe the flow was assigned (**220**). The WFR sets up the required network address translations for locally accepted flows so that future packets within the flow can be modified appropriately (**222**). If the chosen server is "remote" (not in the local server farm) (**220**), an HTTP redirect is generated (**222**) that causes the client to go to the chosen remote site for service.

In addition to the steps described above, which occur as part of the flow setup process, the components shown in FIG. **2** perform several other tasks, including the following. Periodically, the ICP probes the servers **100***a–c* front-ended by the content-aware flow switch **110** for information regarding server status and content. This activity may be undertaken proactively (such as polling for general server health) or at the request of the CSD. The ICP updates the CSD with the results of this search so that future requests for the same content will receive better service (**224**).

The IPP periodically sends local server load and content information to neighboring content-aware flow switches. Data arriving from these peers is evaluated and appropriate updates are sent to the CSD (**226**). The IPA periodically updates the CSD with internet proximity information (**228**).

The operation of the components shown in FIG. 2 is now described in more detail.

Referring to FIG. **3**, the WFR services a client content request as follows. When a client sends a content request to a server in the form of a TCP SYN or HTTP GET, the content request is intercepted by the content-aware flow switch **110**, which interprets the request as a request to initiate a flow between the client and an appropriate server (step **402**). The CSD is queried for a list of available servers to serve the content request (step **404**). The CSD returns a list of candidate servers and the status indicator ACCEPT if the preferred server is known to be in the local server farm. If the CSD returns a status indicator ACCEPT (decision step **406**), then the content request may be served at one of the local servers **100***a–c* front-ended by the flow switch **110**. In this case, the FAC is asked to assign a flow for servicing the content request to a local server, chosen from among the list of candidate servers returned by the CSD (step **408**). If the FAC successfully assigns the flow to a local server (decision step **412**), then an appropriate network address translation for the flow is set up (step **416**), a connection is set up with the appropriate server (using a pre-cached, persistent, or newly created connection) (step **426**), and the content request is passed to the server (step **428**).

If the CSD is unable to identify any local servers to serve the content request (decision step **406**), or if the FAC is unable to assign a flow for the content request to a local server (decision step **412**), then if the status indicator (returned by either the CSD in step **404** or the FAC in step **408**) indicates that the flow should be redirected to a remote server (step **410**), then the flow is redirected to a remote server (step **414**). If the CSD indicated (in step **404**) that the flow should be spoofed (decision step **418**), then the client TCP request is spoofed (step **420**). If the flow cannot be assigned to any server, then the flow is rejected with an appropriate error (step **422**).

Referring to FIG. **4**, the CSD parses a flow setup request as follows. First, the CSD parses the URI representing the client content request in order to identify the nature of the requested content (step **429**). If the request is an HTTP request, for example, elements of the HTTP header, including the HTTP content-type, are extracted. In the case of a non-HTTP request, the combination of protocol number and source/destination port are used to identify the nature of the requested content. In the case of an HTTP request, the content-type or filename extension is used to deduce a QoS class, delay, minimum bandwidth, and frame loss ratio as shown in Table 1, below. The content-size is used to determine the size of the requested flow. Overall flow intensity is monitored by the content-aware flow switch **110** by calculating the average throughput of all flows. The degree to which a particular piece of content served by a server is "hot content" is measured by monitoring the number of hits (requests) the content receives. The burstiness of a flow is determined by calculating the number of flows per content per time unit.

Identifying the nature of the requested content also involves deducing, from the content request and information stored in the CSD, the QoS requirements of the requested content. These QoS requirements include:

Bandwidth, defined by the number of bytes of content to be transferred over the average flow duration.

Delay, defined as the maximum delay suitable for retrieving particular content.

Frame Loss Ratio, defined as the maximum acceptable percentage of frame loss tolerated by the particular type of content.

A QoS class is assigned to a flow based on the flow's calculated QoS requirements. Eight QoS classes are supported by the flow switch **110**. Table 1 indicates how these classes might be used.

TABLE 1

QoS Class	Delay (End to End)	Min Bandwidth	Frame Loss Ratio	Example Applications
0	N/A	N/A	10^{-6}	Control Flows
1	<250 ms	8 KBPS	10^{-8}	Internet Phone
2	Interactive	4 KBPS	10^{-4}	Distance Learning, Telemetry, streaming video/audio
3	500 ms	0–16 Mbps	10^{-4}	Media distribution, multi-user games, interactive TV
4	Low	64 KBPS	Data: 10^{-8} Streaming: 10^{-4}	Entertainment, traditional fax
5	Low	N/A	10^{-8}	Stock Ticker, News
6	N/A	N/A	10^{-6}	Service Distribution, Internet Printing
7	N/A/	N/A	10^{-4}	Best effort traffic (email, Internet fax, database, etc.)

After the nature of the requested content has been identified, the CSD queries its database for records of candidate servers containing the requested content (step **430**). If the CSD cannot find any records in the database to satisfy a given content request (decision step **432**), the ICP/IPP is asked to locate the requested content, in order to increase the probability that future requests for the requested content will be satisfied (step **446**). The CSD then returns a NULL list to the WFR with a status indicator indicating that the flow request should be rejected (steps **434**, **444**).

If one or more matching server records are found (decision step **432**) and the client request is in the form of a HTTP GET (decision step **436**), then the CSD determines whether any of the existing content records exactly matches the requested content (decision step **448**). For example, consider a content request for http://www.company.com/document.html. The CSD will consider a content record for http://www.company.com/* to be an exact match for the content request. The CSD will consider a record for http://www.company.com/ to be a match for the request, but not the most specific match. In the case of an exact match, the CSD sorts the list of candidate servers (identified in step **430**) based on configurable preferences (step **442**). In the case of at least one match but no exact matches, the CSD creates a new record containing default information extracted from the most specific matching record, as well as additional information gleaned from the content request itself (step **450**). This additional information may include the QoS requirements of the flow, based on the port number of the content request, or the filename extension (e.g., ".mpg" might indicate a video clip) contained in the request. The CSD asks the ICP/IPP to probe, in the background, for more specific information to use for future requests (step **452**).

If one or more server records are found (decision step **432**) and the client content request is in the form of a TCP SYN (decision step **436**), the mere receipt by the flow switch of a TCP SYN may not provide the CSD with enough information about the nature of the requested flow for the CSD to make a determination of which available servers can service the requested flow. For example, the TCP SYN may indicate the server to which the content request is addressed, but not indicate which specific piece of content is being requested from the server. If receipt of a HTTP GET from the client is required to identify a server to serve the content request (decision step **438**), then the CSD returns a NULL server list to the WFR with a status indicator requesting that the TCP connection be spoofed and that the subsequent HTTP GET from the client be forwarded to the CSD (step **440**).

If the TCP SYN is adequate to identify a server to service the content request (decision step **438**), then the CSD sorts the list of candidate servers (identified in step **430**) based on configurable preferences (step **442**).

If adequate information was available in the content request to generate a list of available servers (decision step **432**) and the request may be serviced by one of the servers locally attached to the data switch (decision step **451**), then the Client Capability Database (CCD) is queried for any available information on the capabilities of the requesting client (step **453**).

Referring to FIG. **5**, given a content request and a list of candidate servers, the CSD sorts the list of candidate servers as follows. If the CSD content records indicate that the requested content is "sticky" (i.e., that a client who accesses such content must remain attached to a single server for the duration of the transaction between the client and the server, which could be comprised of multiple individual content requests) (decision step **454**), then the CSD searches an internal database to determine to which server this client was previously "stuck" (step 456). If the CSD finds no record for this client (decision step 458), then the CSD indicates that the request should be rejected (step 464). If the CSD finds a record of this client (decision step 458), then the CSD creates and returns a list of candidate servers which includes only the "sticky" server to which the client was previously "stuck" (step 460), and indicates that a local server to serve the content request was found (step 462). If the requested content is not "sticky" (decision step 454), then the list of candidate servers is ordered according to the method of FIG. 6 (step 456).

Referring to FIG. 6, the CSD orders the list of candidate servers as follows. The CSD evaluates the requested content according to several criteria (step 468). The CSD filters the candidate server list and orders (sorts) the candidate servers remaining in the candidate server list (step 470). Servers in the candidate server list are assigned proximity preferences (step 472).

If the first server in the sorted list of candidate servers is a remote server (decision step 474), then the CSD assigns a value of REDIRECT to a status indicator (step 476). If the first server in the sorted list of candidate servers is a local server (decision step 474), then the CSD assigns a value of ACCEPT to the status indicator (step 478). The CSD returns the status indicator and the ordered list of candidate servers (step 480).

Referring to FIG. 7, a particular requested content is evaluated by the CSD as follows. A variable requestFlag is used to store several flags (values which can be either true or false) relating to the requested content. Flags stored in requestFlag include BURSTY (indicating whether the requested content is undergoing a burst of requests), LONG (indicating that this the request is likely to result in a long-lived flow), FREQUENT (indicating that the requested content is frequently requested), and HI_PRIORITY (indicating that the requested content is high priority content).

If the current time at which the requested content is being requested minus the previous time at which the requested content was requested is not greater than avgInterval (the average period of time between flow requests for the requested content) (decision step 482), then a variable burstLength is assigned a value of zero (step 484) and requestFlag is assigned a value of zero (step 486). Otherwise (decision step 482), the value of the variable burstLength is incremented (step 488), and if the value of burstLength is greater than MIN_BURST_RUN (decision step 490), then avgInterval is recalculated (step 492), and the variable requestFlag is assigned a value of BURSTY (step 494). MIN_BURST_RUN is a configurable value which indicates how many sub-avgInterval requests for a given piece of content constitute the beginning of a burst.

A variable runTime is set equal to the current time (step 496). A flag requestFlag is used to store several pieces of information describing the requested content. If the size of the requested content is greater than a predetermined constant SMALL_CONTENT (decision step 498), then the LONG flag in requestFlag is set (step 502). If the requested content is streamed (decision step 500), then the LONG flag in requestFlag is set (step 502). If the number of hits the requested content has received is greater than a predetermined constant HOT_CONTENT (decision step 504), then the FREQUENT flag in requestFlag is set (step 506). If the requested content has previously been flagged as HIGH_PRIORITY (decision step 508), then the HI_PRIORITY flag in requestFlag is set (step 510).

Referring to FIG. 8, the CSD assigns status indicators to the servers in the candidate server list as follows. The first server in the candidate server list is selected (step 514). If the selected server should be filtered (decision step 516), then the selected server is removed from the candidate server list (step 518). Otherwise, the server is evaluated (step 520), and ordering rules are applied to the selected server to assign a status indicator to the selected server (step 522). If there are more servers in the candidate server list (decision step 524), then the next server in the candidate server list is selected (step 526), and steps 516–524 are repeated. Otherwise, assignment of status indicators to the servers in the candidate server list is complete (step 528).

Referring to FIG. 9, servers are filtered from the candidate server list as follows. If a server has not responded to recent queries (decision step 530), is no longer reachable due to a network topology change (decision step 532), or no longer contains the requested content (indicated by an HTTP 404 error in response to a request for the requested content), then the server is flag for removal from the candidate server list (step 536).

Referring to FIG. 10, a server in the candidate server list is evaluated as follows. A variable serverFlag is used to store several flags relating to the server. Flags stored in serverFlag include RECENT_THIS (indicating that a request was recently made to the server for the same content as is being requested by the current content request), RECENT_OTHER (indicating that a request was recently made to the server for content other than the content being requested by the current content request), RECENT_MANY (indicating that many distinct requests for content have recently been made to the server), LOW_BUFFERS (set to TRUE when one or more recent requests have been streamed), RECENT_LONG (indicating that one or more of the server's recent flows was long-lived), LOW_PORT_BW (indicating that the server's port bandwidth is low), and LOW_CACHE (indicating that the server is low on cache resources).

If the server was not recently accessed (decision step 540), then none of the flags in serverFlag are set, and evaluation of the server is complete (step 570). Otherwise, if the server was recently accessed for the same content as is being requested by the current content request (decision step 542), then serverFlag is assigned a value of RECENT_THIS (step 546); otherwise, serverFlag is assigned a value of RECENT_OTHER (step 548). If there have been many recent distinct requests to the server (decision step 550), then the RECENT_MANY flag in serverFlag is set (step 552). If any of the recent requests to the server were streamed (decision step 554), then the LOW_BUFFERS flag of serverFlag is set (step 556). If any of the recent requests to the server were long-lived (decision step 558), then the RECENT_LONG flag of serverFlag is set (step 560). If the port bandwidth of the server is low (decision step 562), then the LOW PORT_BW flag of serverFlag is set (step 564). If the RECENT_OTHER flag of serverFlag is set (decision step 566), then the LOW_CACHE flag of serverFlag is set (step 568).

Referring to FIG. 11, a server in the candidate server list is ordered within the candidate server list as follows. A variable Status is used to indicate whether the server should be placed at the bottom of the candidate server list. Specifically, if the HI_PRIORITY flag of requestFlag is set (decision step 572), then Status is assigned a value according to FIG. 12 (step 574). If the BURSTY flag of requestFlag is set (decision step 576), then Status is assigned a value according to FIG. 13 (step 578). If the FREQUENT flag of requestFlag is set (decision step 580), then Status is assigned a value according to FIG. 14 (step 582). If the LONG flag of requestFlag is set (decision step 584), then Status assigned a value according to FIG. 15 (step 586); otherwise, Status is assigned a value according to FIG. 16 (step 588). If the value of Status is not OKAY (decision step 590), then the server is considered not optimal and is placed at the bottom of the candidate server list (step 584). Otherwise, the server is considered adequate and is not moved within the candidate server list (step 592).

Referring to FIG. 12, in the case of a request for a flow for which the HI PRIORITY flag of requestFlag is set, if the LOW_CACHE flag of serverFlag is set (decision step 596), the RECENT_OTHER flag of serverFlag is set (decision step 598), the LOW_PORT_BW flag of serverFlag is set (decision step 600), or the RECENT_LONG flag of serverFlag is set (decision step 602), then Status is assigned a value of NOT_OPTIMAL (step 608). Otherwise, Status is assigned a value of OKAY (step 604).

Referring to FIG. 13, in the case of a request for a flow for which the BURSTY requestFlag is set and the RECENT_THIS serverFlag is not set (decision step 608), and if either the LOW_CACHE or RECENT_MANY serverFlag is set (decision steps 610 and 612), then Status is assigned a value of NOT_OPTIMAL (step 616). Otherwise, Status is assigned a value of OKAY (step 614).

Referring to FIG. 14, a value is assigned to Status in the case of a request for a flow which is not bursty and not frequently requested as follows. Status is assigned a value of NOT_OPTIMAL (step 644) if any of the following conditions obtain: (1) the LONG flag of requestFlag is set and the LOW_BUFFERS and LOW_CACHE flags of serverFlag are set (decision steps 620, 622, and 624); (2) the RECENT_MANY, RECENT_THIS, and LOW_CACHE flags of serverFlag are set (decision steps 626, 628, and 630); (3) the RECENT_LONG, RECENT_THIS, and LOW_CACHE flags of serverFlag are set (decision steps 632, 634, and 636); or (4) the LONG flag of requestFlag is set and the LOW_PORT_BW flag of serverFlag is set (decision steps 638 and 640). Otherwise, Status is assigned a value of OKAY (step 642).

Referring to FIG. 15, a value is assigned to Status in the case of a request for a flow which is non-bursty, frequently requested, and short-lived as follows. Status is assigned a value of NOT_OPTIMAL (step 664) if any of the following conditions obtain: (1) the LOW_BUFFERS and LOW_CACHE flags of serverFlag are set (decision steps 646, 648); (2) the RECENT_LONG, RECENT_OTHER, and LOW_CACHE flags of serverFlag are set (decision steps 650, 652, and 654); or (3) the RECENT_MANY, RECENT_OTHER, and LOW_CACHE flags of serverFlag are set (decision steps 656, 658, and 660). Otherwise, Status is assigned a value of OKAY (step 662).

Referring to FIG. 16, a value is assigned to Status in the case of request for flows which are not handled by any of FIGS. 12–15 as follows. Status is assigned a value of NOT_OPTIMAL (step 680) if any of the following conditions obtain: (1) the LOW_BUFFERS and LOW_CACHE flags of serverFlag are set (decision steps 666, 668); (2) the RECENT_MANY and LOW_CACHE flags of serverFlag are set (decision steps 67 and 672); or (3) the RECENT_LONG and LOW_PORT_BW flags of serverFlag are set (decision steps 674 and 676). Otherwise, Status is assigned a value of OKAY (step 678).

Referring again to FIG. 6, the servers remaining in the candidate server list are sorted again, this time by proximity to the client making the content request (step 472). The details of sorting by proximity are discussed in more detail below with respect to the Internet Proximity Assist (IPA) and with respect to FIG. 22.

The first server in the candidate server list is examined, and if it is local to the content-aware flow switch 110 (decision step 474), then a variable Status is assigned a value of ACCEPT (step 478), indicating that the content-aware flow switch 110 can service the requested flow using a local server. Otherwise, Status is assigned a value of REDIRECT (step 476), indicating that the flow request should be redirected to a remote server.

The process of deciding whether to create a flow in response to a client content request is referred to as Flow Admission Control (FAC). Referring again to FIG.3, if the value of Status is ACCEPT (decision step 406), then the FAC is asked to assign the requested flow to a local server (step 408). The FAC admits flows into the flow switch 110 based on flow QoS requirements and the amount of link bandwidth, flow switch bandwidth, and flow switch buffers. Flow admission control is performed for each content request in order to verify that adequate resources exist to service the content request, and to offer the content request the level of service indicated by its QoS requirements. If sufficient resources are not available, the content request may be redirected to another site capable of servicing the request or simply be rejected.

More specifically, referring to FIG. 17, the FAC assigns a flow to a local server from among an ordered list of candidate servers, in response to a content request, as follows. First, the FAC fetches the first server record from the list of candidate servers (step 684). If the server record is for a local server (decision step 686), and the local server can satisfy the content request (decision step 690), then the FAC indicates that the content request has been successfully assigned to a local server (step 694). If the server record is not for a local server (decision step 686), then the FAC indicates that the content request should be redirected (step 688).

If the server record is for a local server (decision step 686) that cannot satisfy the content request (decision step 690), and there are more records in the list of candidate servers to evaluate (decision step 696), then the FAC evaluates the next record in the list of candidate servers (step 698) as described above. If all of the records have been evaluated without redirecting the request or assigning the request to a local server, then the content request is rejected, and no flow is set up for the content request (step 700).

Referring to FIG. 18, the FAC attempts to establish a flow between a client and a candidate server, in response to a client content request, as follows. The FAC extracts, from the CSD server record for the candidate server, the egress port of the flow switch to which the candidate server is connected. The FAC also extracts, from the content request, the ingress port of the flow switch at which the content request arrived (step 726). Using the information obtained in step 726 and other information from the candidate server record, the FAC constructs one or more QoS tags (step 728). A QoS tag encapsulates information about the deduced QoS requirements of an existing or requested flow.

If the requested content is not served by a (physical or virtual) web host associated with a flow pipe (decision step 730), then the FAC attempts to add the requested flow to an existing VC pipe (step 732). A VC pipe is a logical aggregation of flows sharing similar characteristics; more specifically, all of the flows aggregated within a single VC pipe share the same ingress port, egress port, and QoS requirements. Otherwise, the FAC attempts to add the requested flow to the flow pipe associated with the server identified by the candidate server record (step **734**). Once the QoS requirements of a flow have been calculated, they are stored in a QoS tag, so that they may be subsequently accessed without needing to be recalculated.

Referring to FIG. **19**, the FAC constructs a QoS tag from a candidate server record, ingress and egress port information, and any available client information, as follows. If the requested content is not to be delivered using TCP (decision step **738**), then the FAC calculates the minimum bandwidth requirement MinBW of the requested content based on the total bandwidth PortBW available to he logical egress port of the flow and the hop latency hopLatency (a static value contained in the candidate server record) of the flow, using the formula:

MinBW=framesize/hopLatency) Formula 1

(step **756**). If the requested content is to be delivered using TCP (decision step **738**), then the FAC calculates the average bandwidth requirement AvgBW of the requested flow based on the size of the candidate server's cache CacheSize (contained in the candidate server record), the TCP window size TcpW (contained in the content request), and the round trip time RTT (determined during the initial flow handshake), using the formula:

AvgBW=min(CacheSize, TcpW)/RTT Formula 2

(step **740**). The FAC uses the average bandwidth AvgBW and the flow switch latency (a constant) to determine the minimum bandwidth requirement MinBW of the requested content using the formula:

MinBW=min(AvgBW*MinToAvg, clientBW) Formula 3

In Formula 3, MinToAvg is the flow switch latency and clientBW is derived from the maximum segment size (MSS) option of the flow request (step **742**).

The content-aware flow switch **110** reserves a fixed amount of buffer space for flows. The FAC is responsible for calculating the buffer requirements (stored in the variable Buffers) of both TCP and non-TCP flows, as follows. If the requested flow is not to be streamed (decision step **744**), then the flow is provided with a best-effort level of buffers (step **758**). Streaming is typically used to deliver real-time audio or video, where a minimum amount of information must be delivered per unit of time. If the content is to be streamed (decision step **744**), then the burst tolerance btol of the flow is calculated (step **746**), the peak bandwidth of the flow is calculated (step **748**), and the buffer requirements of the flow are calculated (step **750**). A QoS tag is constructed containing information derived from the calculated minimum bandwidth requirement and buffer requirements (step **752**). The FAC searches for any other similar existing QoS tags that sufficiently describe the QoS requirements of the requested content (step **754**).

Referring to FIG. **20**, the FAC locates any existing QoS tags which are similar enough (in MinBW and Buffers) to the QoS tag constructed in FIG. **19** to be acceptable for this content request, as follows. If the requested content is not to be delivered via TCP (decision step **764**), then the FAC finds all QoS tags with a higher minimum bandwidth requirement but with lower buffer requirements than the given QoS tag (step **766**). If the content is to be delivered via TCP (decision step **764**), then the FAC finds all QoS tags with a lower minimum bandwidth requirement and higher buffer requirements than the given QoS tag (step **768**). If the requested content is not to be streamed (decision step **770**), then for each existing QoS tag, the FAC calculates the average bandwidth, calculates the TCP window size as TcpW=AvgBW*RTT, and verifies that the TCP window size is at least 4K (the minimum requirement for HTTP transfers) (step **774**). If the requested content is to be streamed (decision step **770**), then the FAC examines each existing QoS tag and excludes those that are not capable of delivering the required peak bandwidth PeakBW or burst tolerance btol, as calculated in FIG. **19**, steps **746** and **748** (step **772**). The resulting list of QoS tags is then used when aggregating the flow into a VC-pipe or flow pipe.

One of the effects of the procedures shown in FIGS. **3–20** is that the flow switch **110** functions as a network address translation device. In this role, it receives TCP session setup requests from clients, terminates those requests on behalf of the servers, and initiates (or reuses) TCP connections to the best-fit target server on the client's behalf. For that reason, two separate TCP sessions exist, one between the client and the flow switch, the other between the flow switch and the best-fit server. As such, the IP, TCP, and possible content headers on packets moving bidirectionally between the client and server are modified as necessary as they traverse the content-aware flow switch **110**.

Flow Pipes

A content-aware flow switch can be used to front-end many web servers. For example, referring to FIG. 1*c*. the flow switch **110** front-ends web servers **100*a–c***. Each of the physical web servers **100*a–c*** may embody one or more virtual web hosts (VWH's). Associated with each of the VWH's front-ended by the flow switch **110** may be a "flow pipe," which is a logical aggregation of the VWH's flows. Flow pipes guarantee an individual VWH a configurable amount of bandwidth through the content-aware flow switch **110**.

Referring to FIG. **21*a***, web servers **100*a–c*** provide service to VWHs **100*d–f*** as follows. Web server **100*a*** provides all services to VWH **100*d***. Web server **100*b*** provides service to VWH **100*e*** and a portion of the services to VWH **100*f***. Web server **100*c*** provides service to the remainder of VWH **100*f***. Associated with VWHs **100*d–f*** are flow pipes **784*a***, **784*b***, and **784*c***, respectively. Note that flow pipes **784*a–c*** are logical entities and are therefore not shown in FIG. **21*a*** as connecting to VWH's **100*d–f*** or the flow switch **110** at physical ports.

The properties of each of the VWH's **100*d–f*** is configured by the system administrator. For example, each of the VWH's **100*d–f*** has a bandwidth reservation. The flow switch **110** uses the bandwidth reservation of a VWH to determine the bandwidth to be reserved for the flow pipe associated with the VWH. The total bandwidth reserved by the flow switch **110** for use by flow pipes, referred to as the flow pipe bandwidth, is the sum of all the individual flow pipe reservations. The flow switch **110** allocates the flow pipe bandwidth and shares it among the individual flow pipes **784*a–c*** using a weighted round robin scheduling algorithm in which the weight assigned to an individual flow pipe is a percentage of the overall bandwidth available to clients. The flow switch **110** guarantees that the average total bandwidth actually available to the flow pipe at any given time is not less than the bandwidth configured for the flow pipe regardless of the other activity in the flow switch **110** at the time. Individual flows within a flow pipe are separately weighted based on their QoS requirements. The flow switch **110** maintains this bandwidth guarantee by proportionally adjusting the weights of the individual flows in the flow pipe so that the sum of the weights remains constant. By policing against over-allocation of bandwidth to a particular VWH, fairness can be achieved among the VWH's competing for outbound bandwidth through the flow switch **110**.

Again referring to FIG. **21***a*, consider the case in which the flow switch **110** is configured to provide service to three VWH's **100***d–f*. Suppose that the bandwidth requirements of VWH **100***d–f* are 64 Kbps, 256 Kbps, and 1.5 Mbps, respectively. The total flow pipe bandwidth reserved by the flow switch **110** is therefore 1.82 Mbps. Assume for purposes of this example that the flow switch **110** is connected to the Internet by uplinks **115***a–c* with bandwidths of 45 Mbps, 1.5 Mbps, and 1.5 Mbps, respectively, providing a total of 48 Mbps of bandwidth to clients. In this example, flow pipe **784***a* is assigned a weight of 0.0013 (64 Kbps/48 Mbps), flow pipe **784***b* is assigned a weight of 0.0053 (256 Kbps/48 Mbps), and flow pipe **784***c* is assigned a weight of 0.0312 (1.5 Mbps/48 Mbps). As individual flows within flow pipes **784***a–c* are created and destroyed, the weights of the individual flows are adjusted such that the total weight of the flow pipe is held constant.

The relationship between flows, flow pipes, and the physical ingress ports **170***a–c* and physical egress ports **165***a–c* of the content-aware flow switch **110** is discussed below in connection with FIG. **21***b*. Flows **782***a–c* from VWH **100***d* enter the flow switch at egress port **165***a*. Flows **786***a–b* from VWH **100***c* enter the flow switch at egress port **165***b*. Flow **786***c* from VWH loof enters the flow switch at egress port **165***b*. Flows **788***a–c* from VWH loof enters the flow switch from egress port **165***c*. After entering the flow switch **110**, the flows **782***a–c*, **786***a–c*, and **788***a–c* are managed within their respective flow pipes **784***a–c* as they pass through the switching matrix **790**. The switching matrix is a logical entity that associates a logical ingress port and a logical egress port with each of the flows **782***a–c*, **786***a–c*, and **788***a–c*. As previously mentioned, each of the physical ingress ports **170***a–c* may act as one or more logical ingress ports, and each of the physical egress ports **165***a–c* may act as one or more logical egress ports. FIG. **21***b* shows a possible set of associations of physical ingress ports with flow pipes and physical egress ports for the flows **782***a–c*, **786***a–c*, and **788***a–c*.

Internet Proximity Assist

A client may request content that is available from several candidate servers. In such a case, the Internet Proximity Assist (IPA) module of the content-aware flow switch **110** assigns a preference to servers which are determined to be "closest" to the client, as follows.

The Internet is composed of a number of independent Autonomous Systems (AS's). An Autonomous System is a collection of networks under a single administrative authority, typically an Internet Service Provider (ISP). The ISPs are organized into a loose hierarchy. A small number of "backbone" ISPs exist at the top of the hierarchy. Multiple AS's may be assigned to each backbone service provider. Backbone service providers exchange network traffic at Network Access Points (NAPs). Therefore, network congestion is more likely to occur when a data stream must pass through one or more NAPs from the client to the server. The IPA module of the content-aware flow switch **110** attempts to decrease the number of NAPs between a client and a server by making an appropriate choice of server.

The IPA uses a continental proximity lookup table which associates IP addresses with continents as follows. Most IP address ranges are allocated to continental registries. The registries, in turn, allocate each of the address ranges to entities within a particular continent. The continental proximity lookup table may be implemented using a Patricia tree which is built based on the IP address ranges that have been allocated to various continental registries. The tree can then be searched using the well-known Patricia search algorithm. An IP address is used as a search key. The search results in a continent code, which is an integer value that represents the continent to which the address is registered. Given the current allocations of IP addresses, the possible return values are shown in Table 2.

TABLE 2

ID	Continent
0	Unknown
1	Europe
2	North America
3	Central and South America
4	Pacific Rim

Additional return values can be added as IP addresses are allocated to new continental registries. Given the current allocation of addresses, the continental proximity table used by the IPA is shown in Table 3.

IP ADDRESS RANGE	CONTINENT IDENTIFIER
0.0.0.0 through 192.255.255.255	0 (Unknown)
193.0.0.0 through 195.255.255.255	1 (Europe)
196.0.0.0 through 197.255.255.255	0 (Unknown)
198.0.0.0 through 199.255.255.255	2 (North America)
200.0.0.0 through 201.255.255.255	3 (Central and South America)
202.0.0.0 through 203.255.255.255	4 (Pacific Rim)
204.0.0.0 through 209.255.255.255	2 (North America)
210.0.0.0 through 211.255.255.255	4 (Pacific Rim)
212.0.0.0 through 223.255.255.255	0 (Unknown)

Referring to FIG. **22**, the IPA assigns proximity preferences to zero or more servers, from a list of candidate servers and a client content request, as follows. The IPA identifies the continental location of the client (step **800**). If the client continent is not known (decision step **801**), then control passes to step **812**, described below. Otherwise, the IPA identifies the continental location of each of the candidate servers (step **802**) using the continental proximity lookup table, described above. If all of the server continents are unknown (decision step **803**), control passes to step **807**, described below. Otherwise, if none of the candidate servers are in the same continent as the client (decision step **804**), then the IPA does not assign a proximity preference to any of the candidate servers (step **806**).

At step **807**, the IPA prunes the list of candidate servers to those which are either unknown or in the same continent as the client. If there is exactly one server in the same continent as the client (decision step **808**), then the server in the same continent as the client is assigned a proximity preference (decision step **810**). For purposes of decision steps **804** and **808**, a client and a server are considered to reside in the same continent if their lookup results match and the matching value is not 0 (unknown).

If there is more than one server in the same continent as the client (decision step **808**), then the IPA assigns a proximity preference to one or more servers, if any, which share a "closest" backbone ISP with the client, where "closest" means that the backbone ISP can reach the client without going through another backbone ISP. A closest-backbone lookup table, which may be implemented using a Patricia tree, stores information about which backbone AS's are closest to each range of IP addresses. An IP address is used as the key for a search in the closest-backbone lookup table. The result of a search is a possibly empty list of AS's which are closest to the IP address used as a search key.

The IPA performs a query on the closest-backbone lookup table using the client's IP address to obtain a possibly empty list of the AS's that are closest to the client (step **812**). The IPA queries the closest-backbone lookup table to obtain the AS's which are closest to each of the candidate servers previously identified as being in the same continent as the client (step **814**). The IPA then identifies all candidate servers whose query results contain an AS that belongs to the same ISP as any AS resulting from the client query performed in step **812** (step **816**). Each of the servers identified in step **816** is then assigned a proximity preference (step **818**).

After any proximity preferences have been assigned in either step **810** or **818**, the existence of a network path between the client and each of the preferred servers is verified (step **820**). To verify the existence of a network path between the client and a server, the content-aware flow switch **110** queries the content-aware flow switch that front-ends the server. The remote content-aware flow switch either does a Border Gateway Protocol (BGP) route table lookup or performs a connectivity test, such as by sending a PING packet to the client, to determine whether a network path exists between the client and the server. The remote content-aware flow switch then sends a message to the content-aware flow switch **110** indicating whether such a path exists. Any server for which the existence of a network path cannot be verified is not assigned a proximity preference. Servers to which a proximity preference has been assigned are moved to the top of the candidate server list (step **822**).

Because multiple AS's may be assigned to a single ISP, an ISP-AS lookup table is used to perform step **816**. The ISP-AS lookup table is an array in which each element associates an AS with an ISP. An AS is used as a key to query the table, and the result of a query is the ISP to which the key AS is assigned.

Referring to FIG. **23**, the invention may be implemented in digital electronic circuitry or in computer hardware, firmware, software, or in combinations of them. Apparatus of the invention may be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a computer processor **1080**; and method steps of the invention may be performed by a computer processor **1080** executing a program to perform functions of the invention by operating on input data and generating output. The processor **1080** receives instructions and data from a read-only memory (ROM) **1120** and/or a random access memory (RAM) **1110** through a CPU bus **1100**. The processor **1080** can also receive programs and data from a storage medium such as an internal disk **1030** operating through a mass storage interface **1040** or a removable disk **1010** operating through an I/O interface **1020**. The flow of data over an I/O bus **1050** to and from I/O devices and the processor **1080** and memory **1110, 1120** is controlled by an I/O controller **1090**.

The present invention has been described in terms of an embodiment. The invention, however, is not limited to the embodiment depicted and described. Rather, the scope of the invention is defined by the claims.

What is claimed is:

1. In a network, a method for directing packets between a client and a server, the method comprising:

receiving a client request for content via the network;

deriving, from the client request, content information descriptive of a plurality of characteristics of the content requested by the client request;

in response to receiving the client request, selecting a server from among a set of candidate servers based on

i) the derived content information; and

ii) a combination of server metrics obtained after receipt of the client request from all available servers capable of servicing the client request for content;

subsequently forwarding to the selected server transmissions originating from the client which are associated with the client request for content; and

subsequently forwarding to the client transmissions originating from the selected server which are associated with the client request for content.

2. The method of claim **1**, wherein the client request is an HTTP request.

3. The method of claim **2**, wherein deriving content information comprises:

extracting information from at least one portion of an HTTP header of the client request.

4. The method of claim **2**, wherein deriving content information comprises deriving content information based on a Universal Resource Locator (URL) included in the client request.

5. The method of claim **2**, wherein deriving content information comprises deriving content information based on a filename included in the client request.

6. The method of claim **5**, wherein deriving content information based on a filename comprises deriving content information based on the filename extension.

7. The method of claim **2**, wherein deriving content information comprises deriving content information based on a port identified in the client request.

8. The method of claim **2**, wherein deriving content information comprises deriving content information based on query parameters including in the client request.

9. The method of claim **8**, wherein the query parameters comprise Common Gateway Interface (CGI) parameters included in a URL of the client request.

10. The method of claim **2**, wherein the client request comprises one of the following: an HTTP GET message, an HTTP HEAD message, an HTTP PUT message, and an HTTP POST message.

11. The method of claim **2**, wherein deriving content information comprises extracting information from the body of the client request.

12. The method of claim **1**, wherein the client request is a TCP request.

13. The method of claim **1**, further comprising:

obtaining additional information from the client about the content requested by the client request; and

wherein the selecting further comprises selecting based on the additional information.

14. The method of claim **1**, further comprising:

obtaining client capability information about the client; and

wherein the selecting further comprises selecting the selected server based on the client capability information.

15. The method of claim 1, wherein selecting as the server comprises:

determining whether the client request requires persistent connectivity with a particular candidate server;

if the client request requires persistent connectivity with a particular server, identifying a candidate server with which the client is persistently connected for service of the client request;

selecting the identified candidate server.

16. The method of claim 1, further comprising determining whether an active path exists between the client and the selected server.

17. The method of claim 16, wherein determining whether an active path exists comprises sending a PING packet to the client.

18. The method of claim 16, wherein determining whether an active path exists comprises performing a Border Gateway Protocol route table lookup.

19. The method of claim 16, wherein the location of the client comprises a continent in which the client resides.

20. The method of claim 19, wherein the locations of the plurality of servers are continents in which the servers reside.

21. The method of claim 16, wherein identifying servers that are in the same location as the client comprises:

identifying administrative authorities associated with the client;

identifying, for each of the plurality of servers, administrative authorities associated with the server; and

identifying servers associated with an administrative authority that is associated with the client.

22. The method of claim 21, wherein the administrative authorities are Internet Service Providers.

23. The method of claim 1, further comprising

deriving, from the client request, quality of service information descriptive of quality of service requirements of the content requested by the client request; and

wherein the selecting further comprises selecting based on the quality of service information.

24. The method of claim 1, wherein the deriving quality of service information includes deriving quality of service information from the content information.

25. The method of claim 1, wherein the deriving quality of service information includes deriving quality of service information from a size of the content requested by the client request.

26. The method of claim 1, wherein quality of service requirements comprise a bandwidth.

27. The method of claim 1, wherein quality of service requirements comprise a delay.

28. The method of claim 1, wherein quality of service requirements comprise a frame loss ratio.

29. The method of claim 1, wherein deriving quality of service information comprises deriving quality of service information from the MIME content type of the client request.

30. The method of claim 1, wherein deriving information descriptive of the content comprises deriving information descriptive of the content type.

31. The method of claim 1, wherein the selecting further comprises selecting based on at least one server metric describing at least one expected level of service provided by at least one of the candidate servers when serving the requested content.

32. The method of claim 31, wherein the at least one server metric includes:

one or more metrics selected from the following group: a metric descriptive of server availability, a metric descriptive of the current load of at least one of the candidate servers, a metric descriptive of recent activity on at least one of the candidate servers, a metric descriptive of network congestion between the client and at least one of the candidate servers, a metric descriptive of the number of active connections being maintained by at least one of the candidate servers, a metric descriptive of the response time of at least one of the candidate servers, information descriptive of one or more previous selections of candidate servers, and client-server proximity information descriptive of distances between the client and at least one of the candidate servers.

33. The method of claim 32, wherein client-server proximity information comprises information descriptive of a continent in which the client resides and a continent in which the server resides.

34. The method of claim 33, wherein client-server proximity information further comprises information descriptive of an administrative authority associated with the client and an administrative authority associated with the server.

35. The method of claim 34, wherein the administrative authorities are Internet Service Providers.

36. The method of claim 31, wherein the at least one server metric includes:

two or more metrics selected from the following group: a metric descriptive of server availability, a metric descriptive of the current load of at least one of the candidate servers, a metric descriptive of recent activity on at least one of the candidate servers, a metric descriptive of network congestion between the client and at least one of the candidate servers, a metric descriptive of the number of active connections being maintained by at least one of the candidate servers, a metric descriptive of the response time of at least one of the candidate servers, information descriptive of one or more previous selections of candidate servers, and client-server proximity information descriptive of distances between the client and at least one of the candidate servers.

37. The method of claim 31, wherein the at least one server metric is obtained by querying a database.

38. The method of claim 31, wherein the at least one server metric is obtained by periodically querying servers in the Internet Protocol network.

39. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of whether the candidate server is receiving a burst of requests for the content requested by the client request.

40. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of whether satisfying the client request will result in a short-term flow.

41. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of whether the content requested by the client request has been frequently requested in the past.

42. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of whether the content requested by the client request has a high priority.

43. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of a probability that the content requested by the client request is cached by the server.

44. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of whether the candidate server has responded to recent queries.

45. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of whether the candidate server recently responded to a request for the content requested by the client request with an indication that the content is not served by the candidate server.

46. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of whether the candidate server is reachable.

47. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of whether the candidate server's cache resources are below a threshold level.

48. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of whether the candidate server's active network connections are below a threshold level.

49. The method of claim 31, wherein the expected level of service provided by a candidate server is descriptive of whether the candidate server's network bandwidth is below a threshold level.

50. A system for directing a stream of packets between a client and a server, the system comprising:

a plurality of servers;

a switch coupled to the plurality of servers by an Internet Protocol network through one or more communication links, wherein the switch comprises:

means for receiving a client request for content via the Internet Protocol network;

means for deriving, from the client request, content information descriptive of a plurality of characteristics of the content requested by the content request;

means, responsive to the means for deriving, for selecting a server from among a set of candidate servers serving the content requested by the client request, based on the content information;

means for subsequently forwarding to the selected server transmissions originating from the client which are associated with the client request for content; and

means for subsequently forwarding to the client transmissions originating from the selected server which are associated with the client request for content.

51. The system of claim 50, wherein:

the candidate servers comprise HTTP servers.

52. A switch in an Internet Protocol network, comprising:

means for receiving a client request for content via the Internet Protocol network;

means for deriving, form the client request, content information descriptive of a plurality of characteristics of the content requested by the content request;

means, responsive to the means for deriving, for selecting a server from among a set of candidate servers serving the content requested by the client request, based on the content information;

means for subsequently forwarding to the selected server transmissions originating from the client which are associated with the client request for content; and

means for subsequently forwarding to the client transmissions originating from the selected server which are associated with the client request for content.

53. In an Internet Protocol network, a method for use in a network switch, the method directing packets between a client and a server, the method comprising:

receiving an HTTP (HyperText Transfer Protocol) client request for content via the Internet Protocol network at an ingress port of the switch;

determining the content requested by the client request based on a plurality of characteristics related to portions of the client request;

selecting a server from among a set of candidate servers based on the determining;

subsequently forwarding to the selected server packets originating from the client which are associated with the client request for content via a switch egress port; and

subsequently forwarding to the client transmissions originating from the selected server which are associated with the client request for content via a switch egress port.

54. The method of claim 53, wherein the determining comprises determining based on a URL (Universal Resource Locator) included in the request.

55. The method of claim 53, wherein the determining comprises determining based on the requested domain name included in the request.

56. The method of claim 53, wherein the determining comprises determining a type of content requested.

* * * * *